USAISEC

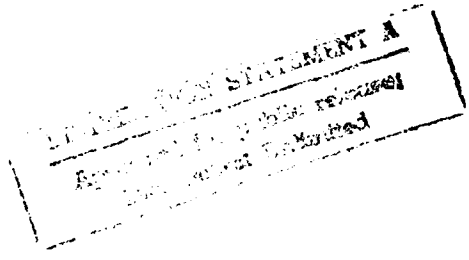
US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES

AIRMICS

Automation of Dynamic Help for U.S. Army Installation-Level Software

ASQB-GM-92-002
October 1991



AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICAION AUTHORITY N/A	3. DISTRIBUTION/AVAILIBILTY OF REPORT N/A
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ASQB-GM-92-002	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION AIRMICS 6b. OFFICE SYMBOL (*If applicable*) ASQB-GM	7a. NAME OF MONITORING ORGANIZATION N/A
6c. ADDRESS (City, State, and Zip Code) 115 O'Keefe Building Georgia Institute of Technology Atlanta, Georgia 30332-0800	7b. ADDRESS (*City, State, and ZIP Code*) N/A
8b. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS 8b. OFFICE SYMBOL (*If applicable*) ASQB-GM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*) 115 O'Keefe Bldg. Georgia Institute of Technology Atlanta, GA 30332-0800	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. 62783A; PROJECT NO. DY10; TASK NO. 05; WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
Automation of Dynamic Help for US Army Installation-Level Software

12. PERSONAL AUTHOR(S)
Cpt. Stanley K. Haines, US Army

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
	FROM_____ TO_____	October 1991	135

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

The purpose of this paper was to determine the extent to which Dynamic Help specification can be automated, and to propose a cost-effective procedure for implementing Dynamic Help for the US Army's Installation Support Modules (ISM) programs. The proposed method decomposes the message into fixed structure sentences having variable slots to be filled with context-specific data. Each slot depends on only one or a few aspects of the context. It is concluded that Dynamic Help is greatly automatable, and a set of programmer's tools can be developed to apply the automation method to provide Dynamic Help messages for many of the Army's software packages.

20. DISTRIBUTIONN/AVAILIBILTY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	
22a. NAME OF RESPONSIBLE INDIVIDUAL LTC Michael E. Mizell	22b. TELEPHONE (*Include Area Code*) (404) 894-3107
	22c. OFFICE SYMBOL ASQB-GM

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

This research was performed for the Army Institute for Research in Management Information, Communications, and Computer Sciences (AIRMICS), the RDTE organization of the U.S. Army Information Systems Engineering Command (USAISEC).This research report is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

s/ James Gantt
James Gantt
Division Chief
MISD

s/ John R. Mitchell
John R. Mitchell
Director
AIRMICS

AUTOMATION OF DYNAMIC HELP FOR U.S. ARMY INSTALLATION-LEVEL SOFTWARE

A THESIS
Presented to
The Academic Faculty

by

Stanley Kirk Haines

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Operations Research

Georgia Institute of Technology
October 1991

Automation of Dynamic Help for
U.S. Army Installation-Level Software

Approved:

Donovan Young, Chairman

S. Manivannan

Roy E. Marsten

Date Approved by Chairman Oct. 1, 1991

Acknowledgements

I can never fully express the appreciation and gratitude I have for the tremendous guidance and counsel I received from my thesis advisor, Dr. Donovan B. Young. I learned more than I anticipated during the last year working on this thesis, and I think Dr. Young may have grayed more than he expected.

I thank the other members of my Thesis Advisory Committee, Dr. Roy Marsten and Dr. S. Mannivannan for their assistance in the development of this thesis.

Special thanks to Mr. John Mitchell and his staff at the Army Institute for Research in Management Information, Communications and Computer Science (AIRMICS) for their assistance, support, and Macintosh during the development of this thesis.

Finally, to Carol Ann who concurrently worked on her thesis in Physical Therapy. We did it honey! And to Stanley Jr., who kept us both sane.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

GLOSSARY

ACRONYM	*Meaning*
ACE	Application Connectivity Engineering
ACIFS	Automated Central Issue Facility System
AIRMICS	Army Institution for Research in Management Information, Communications, and Computer Science
DBMS	Data Base Management System
DIC	Document Identification Code
ETIP	Extended Terminal Interface Program
EUS	Embedded User Support
ILIDB	Installation Level Integrated Data Base
ISM	Installation Support Module
SQL	Standard Query Language
UIC	Unit Identification Code

SUMMARY

Dynamic Help is an online system of context-specific help messages introduced by Dr. Donovan Young in 1990 and claimed to be automatable. A semi-automated set of Dynamic Help messages had been implemented by AIRMICS (Wolven 1991) on an interactive program that operates clothing warehouses, and had been determined via user tests to make the program usable without user training.

The questions for this research were to determine the extent to which Dynamic Help specification can be automated, and to propose a cost-effective procedure for implementing Dynamic Help for the U.S. Army's ISM (Installation Support Module) programs.

An automation approach is proposed, based on identifying each interactive context in terms of the currently active objects and acts. The lowest-level act is designated as the primary act, and the message is intended to aid the user in either performing that act or going to a different context. The proposed method decomposes the message into fixed-structure sentences having variable slots to be filled with context-specific data. Each slot depends on only one or a few aspects of the context. Data to fill slots is supplied directly from the running program or its database, where available; where not available, it is stored in Dynamic Help dictionaries whose contents the designer must author specifically for the program (in lieu of authoring messages themselves).

The proposed method was applied to generate all the Dynamic Help messages for a sample of contexts that occur in using the ACIFS program. These messages were compared to the existing messages. It was found that they appeared comparable in clarity. For the sample of automated messages, it was additionally found that they required less

than half of the authorship effort (based on character counts) than the corresponding semi-automated existing messages; for the whole program, a fivefold decrease in authorship effort was estimated on the basis of the results for the sample.

It is concluded that Dynamic Help is greatly automatable, and it is recommended that a Dynamic Help "generator" — a set of programmer's tools — be developed to apply the automation method to provide Dynamic Help messages for many of the Army's ISMs.

CHAPTER I

INTRODUCTION TO USER INTERFACES

1.1 Introduction

The problem of excessive diversity and uneven design of user interfaces for interactive computer programs is widely documented (Gould, 1988). On Army installations (posts, camps, and stations) there are dozens of interactive software packages in daily use. Many of these packages are installation-unique systems that were developed by the Major Army Command (MACOM). Their development and implementation has not provided for much horizontal and vertical integration, nor for data sharing across functional areas. In fact, many of these data initiatives are redundant and are functionally duplicative. The total number of interactive software packages used daily on all US Army Installations is estimated to be between 150 to 200 (Installation Support Modules, 1990).

Very few of these systems have been developed in accordance with a standard architecture. It is difficult to deploy them to other installations or use them across varying hardware platforms. The US Army Installation Support Modules (ISM) project, a high priority project, has been established to enhance installation management Army-wide by integrating this large family of software into a coherent system (Installation Support Modules, 1990). Some of the goals of the ISM project are to develop an interface that non-ADP users will find easy to use and to develop a standard data encyclopedia/dictionary for use at installation level.

The ISM project is applicable at all Army installations. It will support the sustaining software base and will interface with tactical and strategic systems. The ISM

project will provide installation commanders with an Installation Level Integrated Data Base (ILIDB) and with software to effectively manage daily operations and perform the landlord functions outlined in the U.S. Army Regulation 5-3, Installation Management and Organization. The Installation Software Modules will reduce redundant data entry and duplicative input terminals, serve as input and output mechanisms for the Standard Army Management Information System (STAMIS), interface with STAMIS via the ILIDB, and provide the required capability to share accurate and timely information at various levels of installation management. Together ISM and ILIDB will help meet the information needs of tenant activities and higher command echelons and provide the integration mechanism at installation level for tactical and strategic systems (Installation Support Modules, 1990).

1.2 Novice/Intermittent User Productivity

A problem that faces the ISM project as well as software developers in general is the low productivity of the novice or intermittent user. For an expert user or a user who specializes in a single function and operates only one or two software modules, great deficiencies in user interface design and user support can be tolerated and overcome. The user's familiarization and skill-building is refreshed daily and is amortized over thousands of performances. But novice or intermittent users will not enjoy this continual familiarity with a software package. Instead, each time they use a package, they will have to relearn some of the skills, techniques, and commands that are required to accomplish a task. The problems of novice and intermittent users are compounded when software packages do not share reasonably consistent and easy-to-learn user interfaces.

The severity of this problem is directly experienced by the ISM project in term of the training effort necessary to alleviate it. According to Colonel Wayne Byrd, ISM Project Manager, about $12 million of ISM money is expended annually for user training in

conjunction with the fielding of new or revised software. Much of this effort, which includes having teams of software development personnel stay at a site for several days to teach users how to use newly delivered products, could be avoided if interfaces were well enough designed or documented to be immediately usable.

Users on US Army Installations fall into both categories: the expert or consistent user, and the novice or intermittent user. In addition to initial unfamiliarity with newly fielded products, several other factors can cause a user to behave as a novice or intermittent user. If a user only performs a task periodically (monthly or quarterly reports for example), then the user must relearn the software package each time he or she performs the periodic task. Similarly, user who are temporarily assigned elsewhere, or go on leave, or go to a school for training may come back and find they have forgotten how to execute many of their normal tasks. Finally, a user proficient on a certain package may be transferred to another installation that uses a different package, requiring retraining.

The ISM project will make it possible for one user to perform many functions, and for soldiers to perform some functions themselves without a clerk as a chauffeur. As these functions consolidate, the need for a better and more consistent user support is increased. The expected efficiencies from the Installation Support Module project will be realized only if the software modules have user interfaces that are reasonably consistent, easy to learn, and designed for the needs of novice or intermittent users.

1.3 Software and Interfaces

The 150 to 200 software packages currently being used for daily operations of US Army installations are diverse. Some have been thoroughly developed and tested by software engineers. Others are programs written locally by an automation office to solve an immediate problem on the installation. Still others were written by an individual whose

knowledge of interface design is limited to personal experience. But it is obvious that within a category, and especially across categories, there will be a diverse assortment of user interfaces.

1.4 Improvements to User Interfaces

There are three general ways to improve upon a user interface. The entire software package could be rewritten from the ground up, enforcing a consistency in linguistics, style, and interface. If all software packages were rewritten to a common standard, a user trained in the use of, say, an inprocessing package, could easily learn how to use a supply package. When a user was transferred from on installation to another, there would be no need for retraining on the software packages at the new installation.

The chief drawback to rewriting is expense. To rewrite 150 to 200 diverse packages under a common standard would require tens of man-years in programming time. Not only would the programmers have to rewrite packages, but also have to provide platform conversions to handle the diverse computer hardware at installations. Some small packages could be rewritten fairly quickly and easily, but the large packages, like the Central Issue Facility package or the Education Management package, would require considerable programming time.

An alternative to total rewriting is to improve the user interfaces of some of the programs. In many cases, this would require rewriting of only the user interface portions of the code (or adding these portions if they do not exist). This alternative could provide a consistent interface across packages, but it would also require a great deal of programming time. Programming effort could be reduced by patching a shell over the existing interface, but the shell would be another layer of computer code to process. Performance speed of many of the packages would become unacceptably slow.

This is not to say that a given package should not be totally rewritten or have its interface improved upon. If a package is based on old or archaic programming methods or if its design is poor, then the best solution may be a total rewrite. Similarly, the main portion of code in a package may be good, but the user interface may be poor or not to standard so that rewriting the user interface is the best fix. For most packages, however, what is required is an economically attractive method that with minimum time and effort can improve a package's usability.

Less expensive than rewriting a whole program or rewriting its interface is to add sophisticated online documentation that improves its usability. One approach to doing this is called Embedded User Support (EUS)*. Embedded User Support involves adding dynamic documentation to an existing software package. It does not change or necessarily improve upon the existing user interface. Instead, it guides the user past pitfalls and shortcomings in the interface. EUS does not involve rewriting software. A EUS module can be written and tested in far less time than it takes to conduct a total rewrite of a package or to rewrite the user interface.

1.5 Embedded User Support (EUS)

"EUS" was coined by Donovan Young of Georgia Tech in 1990. Based on his experience in designing user interfaces for complex interactive project scheduling software (Young, 1990a), Dr. Young became convinced that not only can a system be programmed to monitor "where" the user is in its logic and what the data state of the system is, and that such context and state information is enough to pinpoint the user's possible immediate

---

* EUS, pronounced "use" as in "useful", is an acronym for Embedded User Support and is also the first part of the word "eusophia" (ευσοφια), which means "having the quality of being easy to learn", taken from the Greek "eu" (ευ) for easy or well and "sophia" (σοφια) for wisdom or skill.

needs for information, but also, that the programming to accomplish this is largely automatable. The following discussions is largely paraphrased, with permission, from Dr. Young's EUS white paper (Young, 1990b).

It is first necessary to draw a distinction between functional training and software familiarization. A "user-friendly" module is one for which the user can acquire the necessary software familiarization by simply interacting with the module. All software packages, in particular all Army installation support modules, can be made user-friendly. On the other hand, the underlying function being supported (equipment issue, housing requests) may require training. EUS does not teach the user what to do functionally, but provides a user-friendly interface to show how the computer does the function.

The purpose of EUS features is to make a software interface sufficiently user-friendly so that the novice or intermittent user can easily resolve all uncertainties about vocabulary, interaction protocols, software capabilities, available actions, data requirements, and standard expectations merely by continuing to interact with the software. A EUS module can teach the user everything about the software's behavior.

Embedded User Support consists of a combination of one or more of the following features:

- Dynamic Help
- what-if data manipulation
- embedded tutorial
- interaction monitoring

1.5.1 Dynamic Help in EUS

Dynamic Help subsystems display state-specific messages that replace older collections of error messages, status reports, and prompts. A modern interactive system is almost always in an input-ready state, and at any one time there is a narrow range of

appropriate inputs. A short message, usually stating what can now be done by the user, is on display at all times (unless the interface design already provides this). Longer, more detailed Help messages are invoked by user requests. Both kinds of Help messages are dynamic; their content depends on the current state and on data values. The dynamic Help subsystem builds messages as part of each interactive response. Thus, the programmer or designer does not write specific messages, but only dictionaries and grammatical structures. The computer will fill in all the details of the help message.

1.5.2 What-if Data Manipulation in EUS

A module that is not part of an integrated database typically grants the user too much data destroying power. Modification to protect real data and allow manipulation of what-if data can be accomplished by changing the module's file manipulation capabilities.

Modern practice is for the system to load a copy of the relevant data in local disk storage or memory, interact with this, and then perform a verification or confirmation step to test validity of the overall set of transactions before issuing the command to change the real database. Protection for a training or practice session is provided simply by suppressing the final step. The what-if data feature is useful not only for training, but for experienced users who want to explore more than one way of accomplishing a transaction. What-it data manipulation is essential in such packages as pricing in procurement, routing in logistics, or scheduling in planning.

1.5.3 Embedded Tutorials in EUS

An ordinary tutorial for a software package is a computer-aided-instruction (CAI) program consisting of a set of lessons and a user monitoring and evaluation system. Each lesson teaches a set of skills for users of the target software package. The user demonstrates skill via responses consisting of correct interactive data entries. The tutorial

program has its own database, and it calls the target software package (or contains a subset or versions of it) to process user responses.

Embedded tutorials differ from ordinary tutorials in that 1) the user can invoke the tutorial from within the module, and 2) a serial path through the tutorial is not enforced. While working on a job, if a user comes to a place where it is desirable to use an unfamiliar feature, he can invoke the tutorial, use it to learn the feature, and return to the job. Unfortunately, tutorials are very inefficient to program, requiring design work and programming work comparable to that of the program itself.

A newer concept for embedded tutorials is the *Dynamic Tutorial* (Smith, 1991). Within the target software package, a dynamic tutorial consists of a set of task statements, a separate database, and a user monitoring and evaluation system. It does not teach, but relies on user friendliness of the system (perhaps as enhanced by Dynamic Help) to let the user discover how to perform tasks.

A Dynamic Tutorial does not break down a task into steps, teach each step, nor provide diagnostic feedback for the user's performance of each step. Instead, it merely monitors overall task completion. Thus a Dynamic Tutorial has advantages of being efficient to design and implement, of allowing for flexibility in the user's choice and order of steps to accomplish a task, and of having "learning leverage" in that the user can discover more than what the tutorial provider specifically provides. Conversely, a Dynamic Tutorial has the disadvantages of requiring user friendliness and of being less able than ordinary tutorials to provide training in the underlying real-world functions.

1.5.4 Interaction Monitoring in EUS

The final possible feature of EUS systems is interaction monitoring, which has been called "learning management" in the computer training arena. There are two alternative levels of interaction monitoring: record/playback, in which a file of user actions

is built during a session so that the session can be reconstructed; and the history file, in which the sequence of major user actions is archived for later analysis. The former is very useful for preparing and playing back a demonstration session, which can be a good training device. The latter performs data aggregation and some analysis during the session, and is typically used as an input file for a separate statistical analysis program that draws conclusions about efficacy and efficiency of the product or of its user.

1.5.5 The EUS Project

In support of the ISM project, the Embedded User Support project is being conducted at the Army Institute for Research, Management, Information Technology, and Computer Science (AIRMICS) located at The Georgia Institute of Technology. The purpose of incorporating EUS capabilities into software systems is the easing of the learning process for a particular computer task by not requiring a user to recall a large amount of information at once. EUS standards will be developed and will incorporate principles of effective human-computer interface design.

The EUS project first added Dynamic Help to the interface for the Automated Central Issue Facility System (ACIFS) program. This program is DBMS-based (built around a relational database management system), which makes the context and data state of the system very easy to monitor. Captain Renée S. Wolven found that, for the kind of system represented by ACIFS, the Dynamic Help features of EUS were relatively easy to provide. She also found that use of Dynamic Help by novice users significantly reduced the number of dead ends (Wolven, 1991). (A dead end occurs where a user cannot proceed without outside assistance). Additionally, for those users for whom complete error data was available (the "careful" user), the use of Dynamic Help significantly reduced the number of data entry errors. The use of Dynamic Help did not improve the overall user performance times for novice users under the test conditions of Captain Wolven's

experiment. For those data entry fields where the user was required to recall difficult data entry formats from memory, the use of Dynamic Help did improve performance times. Captain Wolven also found that the specification of messages appeared to be mostly automatable; she semi-automated the process and pointed out further automation opportunities.

Work on EUS at AIRMICS is continuing with the conversion of ACIFS to the ACE (Application Connectivity Engineering) programming environment provided by AT&T for UNIX-based programming.

1.6 Dynamic Help

This thesis will focus on the EUS feature Dynamic Help. Dynamic Help is context sensitive (dependent on "where" the user is) that is also dynamic (dependent on the system state). It is not interactive; that is, the help system does not collect and interpret its own user input, nor interpose itself into the functional user interaction cycle. Dynamic Help is implemented by adding a Dynamic Help module to a package. A Dynamic Help module includes:

- A set of generic grammatical structures for help sentences
- A set of dictionaries of
  - commands
  - functions
  - objects
  - formats
  - variables
  - domains
  - protocols
- A state monitoring system (if necessary)

The state monitoring system is distributed among the subprograms where necessary if state changes are not reflected in global variables.

Dynamic help is economically attractive because the computer, not the programmer, writes the help messages using parameters. Help messages are built rather than retrieved. For each aspect of the system state, there is a message phrase that is a combination of fixed and variable text. The message displayed is composed of those message phrases that are relevant for the context, with their variable parts filled with values of monitored state variables and the corresponding description from dictionaries.

For example, in an interactive CPM scheduling program, if the user is in the process of reducing the duration attribute of an activity, the help system knows that the context is that of an intermediate stage within the interactive duration change procedure and it also knows the context relevant aspects of the system state, e.g. which object is active and whether it is already at its minimal duration. Thus, a help message, built by the computer, would be specific for the context and state.

What Dynamic Help does *not* do is improve upon a bad user interface; eliminate traps in the software (where a restart is necessary); or correct poor design in the program. These flaws in the software can generally only be corrected by one of the other methods (total rewrite, interface rewrite) described above. Instead, Dynamic Help would help the user understand a confusing or bad interface, tell the user how to avoid traps, or help the user understand what the software requires.

Dynamic Help is more useful for the novice and intermittent user than for the expert user. Regardless of a package's design, interface, or application, an advanced user will have few problems working with a package and around its shortcomings. It is the novice or intermittent user who will be most confused by poor design and a bad interface. A user may accomplish a task many times, but after a break in use (for leave or temporary duty for

example) may become confused by the software again. Upon encountering confusion, a user can invoke Dynamic Help, which will then build a specific help message that tells the user where he is, what his options are, what the software requires as input, and how to leave and start over or go back.

Dynamic Help promises to be a cost effective fix for existing packages as well as a consistent method to help novice and intermittent users perform their missions (Young, 1990a). Until poorly designed packages can be rewritten or weak interfaces can be improved, Dynamic Help allows users to maximize the usability of existing software packages. It provides a consistent method for getting help from the software about the software.

1.7 The Research Problem

The problem attacked in this thesis is to determine the extent to which Dynamic Help message specification can be automated and to propose a cost-effective procedure for implementing EUS add-ons for those ISM programs that will best benefit from EUS.

1.8 Guide to this Thesis

Chapter II reviews the findings of the EUS project, reviews the literature on user-interface effectiveness and online documentation, and classifies and characterizes the user interfaces of the general population of ISM software.

Chapter III examines the structure and synthesis of Dynamic Help messages as exemplified in the EUS project, examines the potential extent to which appropriate messages can by synthesized automatically, proposes a maximally automatable generic structure for Dynamic Help messages, and proposes a procedure for generating messages with that structure.

Chapter IV presents an experiment to test the message-generating procedure proposed in Chapter III. A partial hypothetical documentation database will be created for the ACIFS program and will be shown to contain only data that would appear in any competent documentation system (for example, it will contain an English-language name for every user action that can be taken). The message-generating procedure will be applied to build messages for selected contexts and states by querying this database. These messages will then be shown to have clarity equivalent to those previously generated for the same contexts and states by the EUS project team. The contexts and states for which the automatic message-generating procedure partially fails will be identified and classified.

Assuming the experiment does show automatability for ACIFS, it will be carried further to test the prospective automation efficiency. It is expected that the total amount of message material to be authored and stored with automated Dynamic Help should be no more that half the amount required for a well designed non-automated or semi-automated system.

Chapter V presents conclusions. It generalizes the application of Dynamic Help results to make conclusions about the whole range of EUS techniques. It proposes a specific selection procedure to identify the ISM programs to which EUS features should be added. It identifies what seem to be the most fruitful prospective directions for further EUS research.

CHAPTER II

LITERATURE SEARCH

2.1 Computer Based Help Facilities

Why does computer software need a Help facility? Claire O'Malley provides an explanation:

> Help exists to answer questions. To decide what to put in help, anticipate the questions users may ask and provide quick, clear answers to them. What kinds of questions can we anticipate from users? Just about any kind: How do I...? When do I...? What if I...? What is a ...? Is it true that...? What is the difference between ...and...? What caused...? What did I do wrong...? Why won't this work...? What else can I do...? (O'Malley, 1986)

What should a Help facility do? According to Patricia Dorazio, "The primary function of an online Help facility is to supply immediate command and/or message assistance that allows the user to complete a task."

As computer software has become more complex and computer hardware more powerful, the number of computer users has increased dramatically. With initial computer programs, typically the users were the programmers. As programmers began to share their programs with other computer users, they provided documentation for these other users. As the base of users grew, more and more users entered the computer age. Documentation alone was no longer sufficient; these users needed a Help facility to guide them through the software.

Initially Help facilities were produced after the software had already been put into use. The Help was developed quickly and was in most cases simply the hardcopy reference manual made available online (Dorazio 1988). These first help facilities were

used much the same way that a user would consult a hardcopy reference manual. The user would look up in the index the command, function, or feature that he needed help with. The index would refer to user to a page number in the reference manual.

The first significant improvement to online Help was command-based Help facilities which enabled the computer user to specify the command, function, or feature before going to the index. For example, to get help about a topic, the user enters at the command prompt:

Command ==» **help topic**

The system responds with a screenful or two of information on the requested topic. (Hurd, 1983)

There are many shortcomings with a command-based Help facility, not the least of which is that the user must look up a command by name for a function he wishes to preform. If the user wanted to "erase" an entry, he might reasonably enter:

Command ==» **help erase**

If the index only contained a listing for "delete" with no cross referencing of synonyms, the user would be no closer to accomplishing his task than before requesting help. Many methods attempted to improve on this deficiency. Besides a cross referencing of synonyms, some help facilities listed other related topics at the end of the help message. While this can help the user navigate to the appropriate help message, it still distracts from the task at hand, and it provides no assistance if the target topic is not related to the one accepted.

A popular variation of the command-based Help facility is the menu-driven Help facility. Here when the user invokes help, he selects a choice from lists of options. Selection is made either by typing the choice (or the corresponding number or letter) or selecting the option with a mouse or cursor (Dorazio 1988). A menu-driven Help facility

may provide the user with a list, eliminating the need to know the exact command, but it still distracts the user from the task at hand. The index has been moved to the front and labelled a table of contents.

Another method of providing the user with help is the prompt-driven Help facility. With this facility, the user is led through a series of menu and prompt screens to accomplish the task. A simple example of a prompt-driven system is the automated tellers used by banks (Hurd 1983). This system requires the user to answer a series of questions at each level. Each level distracts the user from the task at hand.

Some software companies provide an online tutorial as a pseudo Help facility. However,

> The goal of a HELP system should not be to teach users about the system's capabilities and function, but rather to provide quick and immediate access to information about a specific task, command, or message. In other words, HELP should refresh or remind the memory of what it already knows (Dorazio, 1986, from Horton 1990)

Tutorials can provide the user knowledge about the idiosyncrasies of a software package, but this is done through the long process of teaching the user about the commands and functions of the package. While a tutorial is sometimes helpful, it is not a Help facility.

Interactive context-sensitive Help is one of the most recent developments in Help facilities.

> The help you get depends on where you are in the program when you request help. If you are executing a particular command, for instance, you would get help on that command. If an error has just occurred, then an explanation of that error will result (Horton, 1990).

Many context-sensitive help facilities link the context-sensitive help and the online reference manual. When the user requests help, the system opens the online reference manual at the topic that corresponds to the user's context. From the initial help screen, the

user can navigate to other help messages, much as for the menu-driven Help facility described above. (Horton, 1990)

A Natural-Language Query Help facility, also called a Diagnostic Help facility (Horton, 1990), tries to diagnose the user's needs by engaging the user in a dialog. If a user needs online help, the user communicates with the system in his natural language. The system translates the natural-language request of the user into a form consistent with its internal look-up table before delivering a response tailored to the user's level of expertise and context (Dorazio, 1988).

In a broader sense, Help facilities can be forced into two categories: passive and active. In active Help facilities, which include menu-driven, prompt-driven, context-sensitive, and natural-language query facilities, both the user and the Help facility play active roles in providing the required information to the user. The user initiates the Help request and the system acts like a human tutor. In a passive help system, the user must explicitly request help for either a command or a message (Dorazio, 1988).

There is a distinct difference between Help facilities and error messages. In both active and passive Help facilities, the user initiates the message by requesting help. He may enter into a dialog with the system to determine the type of help required (such as the Menu-driven or Natural-Language query Help facilities), but the user initiates the help. Error messages are initiated by the computer due to some invalid command, input, or operation. Many error messages include a "fix" suggestion the end of the error message:

> ==» **copy myfile myfile2**
> The file "myfile" is not in the current directory. Check the spelling of the name or change the directory. (Horton, 1990).

Here the likely corrective actions is a known attribute of the error type, and of course outputting this attribute is more useful to the user than outputting other attributes such as what discrepancy was detected, or even what was the assumed cause. The above

example explains the error to the user, then based on what the programmer (or the writer of the help message) felt was the most likely cause of this error, suggests a solution.

2.2 Dynamic Help and Context-Sensitive Help

Dynamic Help differs from context-sensitive help in two ways: *protocol* and *construction*. The protocol for context-sensitive Help requires the user to invoke Help, to indicate for which object of command the help is being sought, and to exit from Help. The protocol for Dynamic Help, by contrast, is not interactive; the user activates objects or commands before invoking help. This allows the Dynamic Help message to be truly specific to the situation, rather than just being a canned blurb about a given object or command. Note that the Dynamic Help protocol presupposes it is possible to activate commands or objects without invoking them. If the basic interactive procedure has "premature closure" (a keyboard entry or screen touch is acted upon instantly without waiting for the user to press «RETURN» or answer a confirm question), the situation is too vague for Dynamic Help to be useful.

Dynamic Help messages differ from context-sensitive messages not only in protocol, but also in construction. A Dynamic Help message is assembled from fixed and variable components rather than retrieved. This makes it automatable, whereas someone must specifically write each message verbatim in a context-sensitive Help facility.

Dynamic Help is neither interactive nor does it require hundreds of different structures for help messages. It is applicable to programs in which the context is quite narrowly defined by such information as cursor location, highlighted objects, and active procedures or commands. Regardless of whether the user is in the right place , that location in the program has a specific task allocated to it. The Dynamic Help module assembles a message explaining everything about performing that task in the current

context. In a narrowly-defined context, the message needs to contain only three things: If in the wrong place, the user needs instructions on *how to leave this context* to find a more appropriate one to suit his purpose; if in the right place, the user needs either instructions on *how to accomplish* a narrowly defined task or an *explanation of the task.*

Dynamic monitoring of the context and state provides a significant advantage over the Help facilities explained above. Dynamic Help has a means of deriving all of the questions that a user could ask. The user is not required to narrow the scope of his question and use an index or table of contents. Of course, this advantage does not exist for systems where the context is not narrowly defined. For example, a user typing at the end of a document in a word processing program is in a vague context where any of dozens of things could be done; Dynamic Help is not useful for highly general programs such as word processors, or spreadsheets, but only for specific application programs.

Dynamic Help's main advantage over context-sensitive help is not in how well it helps the user, but in the cost of providing it. Context-sensitive Help programmers could create a help message for every location within a program, and these messages could be just as understandable as Dynamic Help messages. However, Dynamic Help offers a way of providing a complete array of context specific, state-specific help messages without requiring each message to be written separately. The automation that allows this is the main subject of this thesis.

In order for Dynamic Help to improve the interface clarity for novice and intermittent users it must address the questions shown in Figure 2-1. The answers to these questions are the basis for construction of the Dynamic Help sentences.

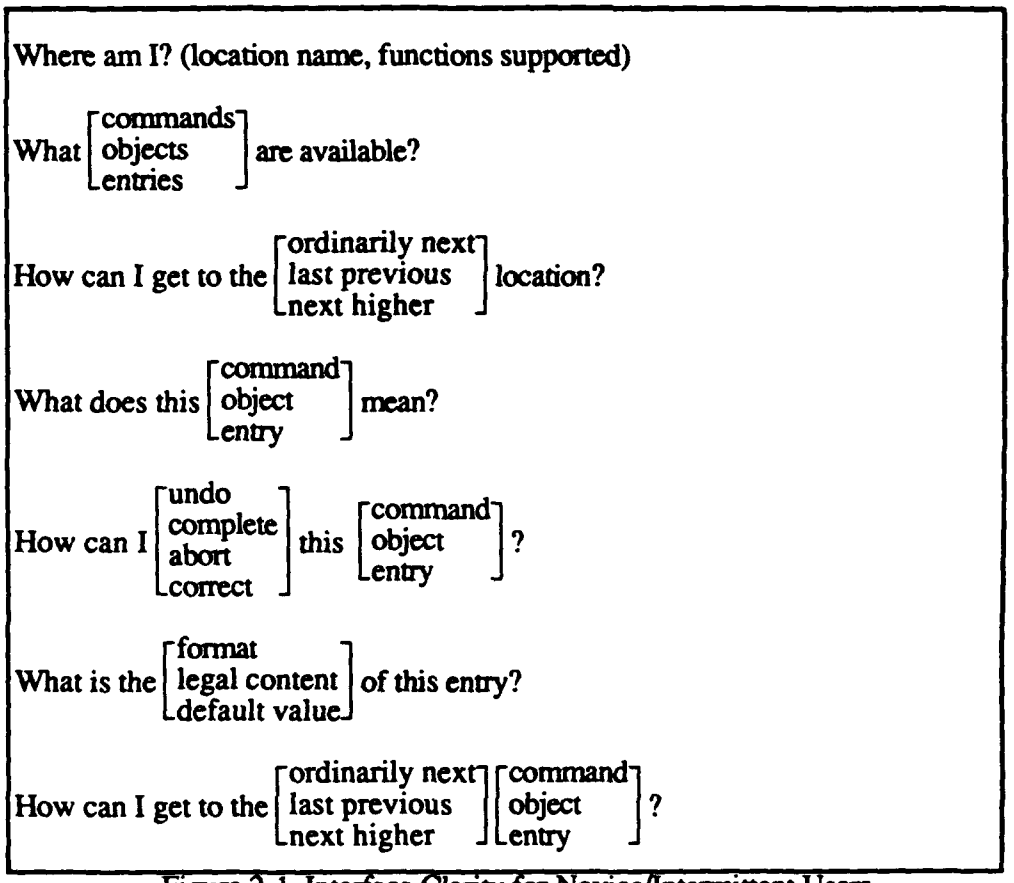


Figure 2-1 Interface Clarity for Novice/Intermittent Users

2.3 Categories of Users

What are the categories of user that require a computer Help facility? Horton (1990) suggests four specific users:

Novice: The Novice knows little about computers and nothing about the program. The Novice is curious, but has trouble distinguishing the essential from the trivial. Most of all, the novice is reluctant to ask for assistance, lacking a vocabulary of concepts and terms to express questions.

Occasional Users: An Occasional user has mastered a system once, but because of intermittent use has forgotten essential items. He makes frequent errors and is impatient with paper documentation. He does not remember computer terms or concepts, nor feels that he should have to. (Cuff, 1980: in Horton, 1990)

Transfer Users: The Transfer user knows how to use one computer system and is trying to transfer this knowledge to a similar system in a new environment.

Experts: The Expert user is commonly referred to as a "Power" user. He understands how to operate the program, how it is organized, and how it works. To him, menus and prompts are obstacles to doing things fast.

John R. Brockman lists five levels of computer user sophistication: parrot, novice, intermediate, expert, and causal (Brockman, 1986). Various authors have advocated separate user interface design levels to tailor interfaces to various kinds of users. Others (Gleason, 1984) have advocated classifying documentation according to sophistication so that users can seek documentation at their own level. However, it is not clear that anyone has presented convincing evidence of the practicality and usefulness of providing more than two levels of documentation or two variations of interactive protocols.

For the purpose of this thesis, I will categorize users into three categories. The Novice user as described above; the Intermittent user which includes the Occasional users and the Transfer user from Horton's definitions; and the Expert user. I view these categories as applying to user familiarity with a specific interface; along with other workers in Dynamic Help, I assume all users have basic familiarity with interactive use of computers in general, and with the real-world function that the interface mediates.

2.4 Software Design

Every interface design has multiple ways of displaying information that helps orient the user, from a symbol that shows there are offscreen contents to which the user can scroll, to full sets of onscreen instructions telling the user exactly what to do. Regardless of what informative strategies are used in the design of a given interface, the interface is likely to have some deficiencies that interfere with usability by a novice or intermittent user. EUS techniques, including Dynamic Help, do not change the interface design. They are added separately.

2.5 Other Issues

The most important characteristic of any help system is the quality of the text presented to the user (Borenstein, 1985). Additionally, in Borenstein's thorough research into various help systems, he reports:

> 1. A good help system can easily make up half the difference between an ordinary bad help system and a human tutor.
>
> 2. The most important determining factor in the "goodness" of a help system seems to be the quality and nature of the texts it presents, rather than the details of the help access mechanism.

Dynamic Help focuses the user on the main task at that level of the program, that is, the primary act. It provides the user with a terse but complete message describing how to complete the current primary act, it explains the meaning, format, domain of any active objects, and it identifies any potential dead-ends.

CHAPTER III

AUTOMATION OF DYNAMIC HELP

To provide for automation of Dynamic Help, this chapter proposes a new model of the context of an interactive session and a new structure of the Dynamic Help message and of each of its sentences. A set of Dynamic Help dictionaries is proposed and is shown to be capable of translating contexts into Dynamic Help messages. The automation method is applied to the ACIFS application program, both to illustrate the method and to provide a sample of messages that can be compared with previous messages for the same contexts in order to test the validity and automation efficiency of the method.

3.1 An Automation Concept for Dynamic Help

From the introduction of Dynamic Help by Dr. Donovan Young (Young, 1990), automation has been an integral part of the concept. The automation concepts and method that I propose and test in this thesis are natural extensions of those developed by Dr. Young, Captain Renée Wolven, Mr. Christopher Smith, and Captain Walter Barge, as reviewed in Chapter II.

The purpose of Dynamic Help automation is to free the designer from having to author hundreds of separate help messages, and to free the program from having to keep help messages ready for retrieval.

The structure of Dynamic Help message, consisting of fixed-structure sentences having variable parts or *slots*, already allows for a substantial degree of what I will call semi-automation: each slot depends not on the whole context but on as few as a single one

of the aspects of the context. For example, one slot in one of the sentences may always call for the name of the current screen, and another slot may always call for the name of the datum that is ready to be added, changed, or deleted. There may be only a few commands or user actions that are used throughout the program, and one slot may always call for a description of the current command or user action. Thus, armed with lists of screens, commands, fields, etc., a designer may take a fill-in-the-blanks approach to authoring messages, and the fixed parts of the messages surrounding the variable slots may be defined as text macros. This approach — call it semi-automation — was taken in the specification of the prototype Dynamic Help messages that were previously implemented in ACIFS and user-tested by Captain Wolven. The goal of the automation method to be proposed here is to provide a substantial decrease in the volume of material authored and stored, as compared to semi-automation.

To achieve this goal, there must be a formalization of context monitoring and identification, and of message construction, sufficient to allow the designer to avoid the necessity to visit every context. A general description of what needs to be formalized follows.

At run time, when the user invokes Dynamic Help, the Dynamic Help module must be able to identify the current context, and it must have a knowledge base that allows it to construct the entire message from the context. This implies a necessity for context monitoring by the program:

> *Context monitoring*: The program monitors the context so that every aspect of the current context is kept current and available to the Dynamic Help module.

The *knowledge base* for the Dynamic Help module must include a context model able to identify the aspects of context that the program is keeping current. It must also include a *message model* that contains the *sentence structures* and definite rules for filing each

variable slot in each sentence. Finally, it must include *Dynamic Help dictionaries* that store the material to fill each variable slot (or store a query or procedure call to return the material). Given the knowledge base, the context identification and message construction functions performed by the Dynamic Help module can be described as follows:

> *Context identification and message construction*: From the message model is obtained the rule for filling each slot, either directly with an aspect of the context, or indirectly with a fragment obtained by querying the Dynamic Help dictionaries to return a fragment that is functionally determined by one or more aspects of the context.

The challenge for the automation method is to achieve the formalizations of context and its aspects, of sentence structures and their variable slots, and of Dynamic Help dictionaries and the slot-specific rules for querying them, so that context monitoring, context identification, and message construction can be performed.

The following three sections will propose the necessary formalizations of context modeling (Section 3.2), and message structure (Sections 3.3) and of Dynamic Help dictionaries (Section 3.4).

3.2 Improvements to Dynamic Help Context Models

An interactive computer session can be perceived in a *user-centered* way, user versus system, where the user takes *actions* that at a low level can be perceived as pressing keys, moving a mouse, etc., or at a higher level can be perceived as making a data entry, highlighting a menu item, pointing at a displayed object, etc. This user-centered viewpoint does not directly focus on the application.

The same session can alternatively be perceived in an *application-centered* way, user and system together as a manipulator of application objects. In this perception there

are *acts* taken by the user and system together, such as issuing a clothing item to a soldier, designating a vehicle as a member of a convoy, etc. This application-centered viewpoint does not directly focus on the interactive interface; it is transparent. This is the viewpoint taken by Dynamic Help.

A successful context model should represent the current acts and objects, because the sentences in Dynamic Help messages turn out to be about them. It is current acts and objects, and their attributes, that fill the message slots. Hence, the "aspects" of the context should be current acts and objets. This was recognized early and was central to the generic automation attempt by Captain Barge (Barge, 1991). Before proposing the context model to be applied and tested here, I will review the previous context models that were used in the prototype ACIFS Dynamic Help system and in Captain Barge's work. Then I will discuss automatability efficiency issues for a hypothetical general context model and for Captain Barge's context model. Finally I will propose a specific context model for Dynamic Help, and will illustrate its treatment of acts, objects, and navigation for the ACIFS program.

3.2.1 Evolution of Dynamic Help Context Modeling

A context is a place or condition in the interaction where at least one aspect of the condition varies enough so that part of a Dynamic Help message for this condition should be different from the corresponding part for another condition. This was the starting point for context modeling for the messages authored and user-tested by Captain Wolven.

This concept implies an operational definition of context: a different context for each different message, and vice versa. One could explore the program as a user, write a message for the current context, take a user action, determine whether the resulting condition is a different context (by examining the entire message for accuracy, informativeness, and completeness), and either record a new context and write a new

message, or expand the description of the current context to include the after-action condition in the same context as the before-action condition.

Given some method of ensuring that all conditions are explored, this process would stop with all contexts defined and all messages written. A method of ensuring completeness of the context set could be based on trying all responses that might change the context, using a branching scheme. Either a breadth-first search or a depth-first search would be equally valid, but a breadth-first search would require less record keeping. A breadth-first search would first try all conditions on the initial screen or window, then all conditions on each screen or window reachable in one step, etc. A depth-first search would "fathom" the software, that is, take a path through various conditions to a condition in which the most detailed or lowest level acts are performed, then "climb back up" to the highest point not yet encountered, and repeat. This depth-first search has the advantage of imitating actual paths taken by users in performing tasks, perhaps allowing the message author to maintain better awareness of the perspective user's knowledge state at each context.

An informed way of organizing a context search would be to begin with a set of tasks to be performed, and imitate their performance. Even if the set of tasks is complete, the set of encountered contexts could be incomplete.

A Dynamic Help designer who used any context-search method for the ACIFS program would find that there are only about 700 reachable contexts. Almost any lowest-level user action — highlighting a new item, making a data entry, pressing a "y" or "n" key to answer a displayed question — causes the program to perform a lowest-level act that alters the context, except when the action is a repeated one on a series of object instances (e.g. entering the quantity of each clothing item being issued to a soldier).

For ACIFS, which contains only menu screens and data-entry screens, the context is neatly indicated by screen identifier and highlight or cursor locations for every lowest-level act, there is a "place" («screen, field») to perform it, and for every "place" there is a unique primary act the user is intended to perform. Only minor exceptions exist to the one-to-one correspondence of «screen, field» to context: different fields in a list of object instances can be the same context (again, e.g. "quantity" fields for each clothing item), and here can be more than one «screen, field» where a given act can be performed (e.g. a menu selection can be made either by highlighting the item and pressing «RETURN» or by pressing the item's code key regardless of which item is highlighted.

Captain Wolven dealt with the exceptions in such a way as to preserve the exact correspondence between «screen, field» and context. Her semi-automated system provided a message for the first field in a list, and then provided a pointer to that message, (in lieu of a copy of it), for other fields in the same list. She also disentangled the menu-selection contexts by defining the highlight-and-RETURN method as the primary act for performing menu selection (a shortcut way of performing an act can either be ignored, since Dynamic Help is for non-expert users, or can be explained in the ad-hoc sentence of messages). Thus the contexts were exactly represented by «screen, field» pairs in the semi-automated set of Wolven messages.

Recall from Chapter II that Captain Barge devised a data organization for Dynamic Help that could be incorporated into the original design of a wide range of application programs (Barge, 1991). He realized that a more general model of contexts would be based not on *screen locations* but on *objects and functions*. Beginning with the same operational concepts as his predecessors — a one-to-one correspondence between messages and contexts, and a primary ("ready to") act for each context — he was led by the wide emphasis in the literature on object-oriented design and function-oriented design to

realize that, in general, a context can be indicated by *combinations of active objects and active functions*. This realization was confirmed when he examined the contexts of the variable parts of the already-established sentence frameworks and found that all of them were attributes of the following objects and functions (Barge, 1991):

Global variables
High-level functions
[data] Objects
Screens (or windows)
Commands
General documentation

The last item in the list, general documentation, refers to context-independent parts of messages. Captain Barge concluded from the list that *dictionaries* of each of the above-listed objects and functions were needed, and that the *context* must track "the current situation in terms of what parts of the system are 'active'." His context identification procedure (Barge, 1991) made use of an implicit assumption that objects and functions were of various kinds or *types*, and that an appropriate design need not allow the complication of a context's having more than one active thing of each type. For example, a university course registration program allows a student to build a class schedule in a schedule window by highlighting courses in a course-offerings window and 'moving' them to and from the schedule. The schedule, which is a set of courses, can be defined as a different type of object than a course, and since the acts are performed one step at a time, there need not be occasion for more than one window, more than one course, or more than one command to be active at any one time. Because it is always possible to decompose a complex act into steps, and to identify collections or subsets separately from the objects that compose them or contain them, the requirement that only one thing of each type can be active in a given context does not seem too restrictive.

In Captain Barge's university registration example, he showed that the context could be unambiguously specified by a vector of four values. For example, the context that yielded the Dynamic Help message

"ready to add ISYE6650A1 as a Primary course ..."

was

Active menu item:	Register for Classes
Active window:	Course Offerings
Active object:	ISYE6650A1
Active command:	P

Dictionaries were built by Captain Barge, and he showed that they produced suitable messages when queried to fill the sentence structures. Although the result was impressive in terms of generality (because it showed that a single Dynamic Help sentence structure applied to two very different programs), the degree of automation was disappointing. One dictionary in particular was a kludge: it filled the "Ready to" sentence with a complete verb clause that expressed a function acting on up to two objects, yet all three variable elements — verb, first object, and second object — were explicitly written rather than retrieved from dictionaries. In such a system there would be a row in the "Ready to" dictionary for every reachable context. Thus one nearly compete "Ready to" sentence would be authored for each message.

The challenge for context modeling can be viewed as that of retaining the generality of Captain Barge's context data organization while achieving a degree of automation in which *no part of the message depends on a large fraction of the context.*

3.2.2 Automatability for a Hypothetical Context Model

Let there be m distinct contexts, so that the number of messages is of order m. If there are c different components in the context vector, and each component can take on d distinct values, then let

$$d^c = m \qquad [1]$$

As an example, for d=10, c=3, we have m=1000.

A hypothetical system might have a context that is identified by its active screen, an active object, and an active command (c is 3 components in the context identifier); if there are 10 screens, 10 objects, and 10 commands (d=10) then equation 1 yields $m=10^3=1000$ contexts.

Automation is possible if the Dynamic Help messages can be divided into parts such that each part depends on only a part of the context. As a concrete example, suppose the hypothetical 1000-context system had messages that could be split into two parts, one of which depended only on the combination of active screen and active object, while the other depended only on the active command. Let a represent the size of the screen/object part of a message, and let 1-a represent the size of the command part of the message, so that the entire message has size 1 whether presented in parts or as a whole. Then if the *same* 10 objects and 10 commands are available on each of the 10 screens, the size of the stored parts is

$$d^2a + d(1-a) = 100a + 10(1-a)$$

This is between 10 and 100, depending on the relative size of the parts. For example, if the screen/object part had size a=2/3 and the command part had size 1-a=1/3, the size of the stored parts would be 70. This compares to a size of 1000 without automation.

Ideally, the best that could be done would be to split the messages into three parts, one of which depended only on the screen, one only on the object, and one only on the command. In this ideal case, supposing each part to have size a=1/3, the size of the stored parts is

$$da + da + da = 10$$

We can recompute the hypothetical example for a more realistic case in which a *different* 10 objects and a *different* 10 commands are available on each of the 10 screens. Then the size of the stored parts is

$$d^3a + d^2(1-a) = 1000a + 100(1-a)$$

Letting the screen/object part have size a=2/3 as before, the size of the stored parts would be 700. Note that in this case there are still 1000 distinct screen/object parts of the message, so the only automation savings is from storing the 100 command parts of the help message separately instead of having them each duplicated in the 10 object-distinct messages for each screen.

In practice, the potential savings from automation of help messages for a typical interactive program would be better than this 700/1000 ratio but worse than the 70/1000 ratio computed earlier. These extremes represent zero redundancy and full redundancy, respectively, of objects and functions on screens, in both cases assuming that 1/3 of each message is redundant to parts of other messages.

3.2.3 Automatability for Barge's Context Model

Examination of Captain Barge's "Ready to" sentences reveals that the context vector, although it contains the minimum set of variables to identify the various conditions, does not always contain the variables needed to fill (either directly or by dictionary reference) its sentences. Table 3-1 extracts from Captain Barge's report (Barge, 1991) six contexts and the corresponding "Ready to" sentences:

Table 3-1 Captain Barge's Contexts and "Ready to" Sentences

Contexts		"Ready to" Sentences
1. Menu Item Window Object Command	Register for Classes None None None	1 Ready to create or modify your academic class schedule.
2. Menu Item Window Object Command	Register for Classes Course Offerings IYSE6650A1 None	2 Ready to choose a command for ISYE6650A1.
3. Menu Item Window Object Command:	Register for Classes Course Offerings IYSE6650A1 P	3 Ready to add ISYE6650A1as a Primary course on your schedule.
4. Menu Item Window Object Command	Register for Classes Schedule None None	4 Ready to modify or finalize your course schedule.
5. Menu Item Window Object Command	Register for Classes Schedule ISYE6650A1 None	5 Ready to choose a command for ISYE6650A1.
6. Menu Item Window Object Command	Register for Classes Schedule ISYE6650A1 R	6 Ready to register for the primary course(s) shown in the Schedule window.

Note that every "Ready to" sentence contains nouns that are not in the context vector; given this, automatability would require that the nouns either be attributes of single items of the context vector or be part of a fixed phrase that itself meets these requirements. Any more complex way of determining noun phrases from contexts would be undesirable. Furthermore, it would be desirable for each slot in a sentence to have an unvarying rule by which given elements of the context vector (only one, where possible) fill the slot. If the slot is not filled directly with a context-vector value, it can be filled indirectly with material retrieved from Dynamic Help dictionaries.

It would be desirable for the context vector to contain all information about the application objects (schedule, course, etc.), and for the dictionaries to contain only information about message objects (sentences, clauses, phrases, etc.).

To judge the automatability of Captain Barge's context model, consider the noun phrase listed in the "Ready to" sentences as "your academic class schedule" (sentence 1), "your schedule" (sentences 3 and 6), and "your course schedule" (sentence 4). Because the messages are largely authored rather than constructed, the schedule object is named several ways; with automation, it would be named only one way in all messages, say "your class schedule," and obviously little or no useful information would be lost thereby. A much more important barrier is that this schedule object is grammatically the direct object of "Ready to" sentences 1 and 4, and the object of a preposition in "Ready to" sentences 3 and 6; in the former sentences it is the thing directly being manipulated, while in the latter sentences it is something indirectly being manipulated via direct manipulation of a lower-level object.

Apparently whenever the *menu item* element of the context vector has the value "Register for Classes", the schedule is "active" and is the object to be *directly* manipulated by an act that is higher level (such as "create or modify" of "modify or finalize"), while it is the object to be *indirectly* manipulated by an act that is lower level (such as "add... as a Primary course" or "remove"). Thus automatability appears to require that objects and acts have *levels*; that more than one, but only one at a given level, can be active; and that the directly manipulated object by an act would be the lowest-level active object, while the indirectly manipulated object would be the next-higher-level active object. This apparently would allow a consistent rule for filling the object slots in at least the "Ready to" sentence.

The apparent fact that the schedule object is active whenever the active menu item is "Register for Classes" could be tracked either in the context vector or in the Dynamic Help

dictionaries. The context vector is the correct place for this, for two reasons: it is fact about application objects, not about messages, and secondly, putting it in the context vector allows flexibility and reliability, because the real rule about when the schedule object is active might be more subtle, and it would be safer to read the object's status rather than assume a rule for it.

.C4.3.2.4 A Proposed Dynamic Help Context Model

As motivated by the preceding automatability analyses of the Wolven context model, a hypothetical context model, and the Barge context model, the following context model is proposed:

1. Let the *acts* be classified by *type*, and let the types have *levels*, so that in a context there can be *up to one active act of each type*, and each active act is of a higher or lower level than any other active act in the same context.

2. Let the *objects* be classified by *type*, and let the types have *levels*, so that in a context there can be *up to one active object of each type*, and each active act is of a higher or lower level than any other active object in the same context.

3. Let the *context vector* be a list of identifiers of active acts of each type and of active objects of each type. If there are N types of acts, let element 1 of the context vector identify the highest-level act type, element 2 identify the next-highest, and so forth, an element N identify the lowest-level act. Then, if there are M types of object, let element $N+1$ identify the active object of the highest-level object type, element $N+2$ identify the next-highest, and so forth, and element $N+M$ identify the lowest-level object.

4. Let the *primary act* be defined as the active act with the lowest act-type level among those levels having an active act (e.g. if the active act for the lowest level is null, the primary act is the active act of the next higher level that is not null).

This context model will allow slots in message sentences to be filled, each according to a slot-specific rule, with any of the values given in Table 3-2 (listed in approximately decreasing order of desirability for achieving simplicity and a high degree of automation):

Table 3-2 Values to be Retrieved to Fill Sentence Slots

Best	The name of the active act or object of a given type
	The name of the active act or object whose level is specified either absolutely or relatively (e.g. the act one level higher than the primary act).
⇕	An attribute (to be retrieved from Dynamic Help dictionaries) of an active act or object (specified by type or level as above).
	An attribute (to be retrieved from Dynamic Help dictionaries) of more than one active act or object.
Worst	An attribute to be retrieved from Dynamic Help dictionaries by a fixed procedure that uses context values as input.

The adequacy of this proposed context model must be judged according to how well it supports the message structure to be proposed below (and, indirectly, according to the performance of constructed messages, to be tested in Chapter IV).

The context vector may be augmented with additional elements of two kinds: a *context key* element and any number of *appositive* elements. The context key is an identifier of the context, which may be useful for design or debugging purposes. It should be noted that such a key would not be useful at run time, since the purpose of Dynamic Help automation is to materialize contexts at run time so as to avoid having to store and retrieve them; but a hypothetical table of contexts, having a distinct context vector as each

row, would be a definition have *all* the context-defining elements as a multiple key, so that a context identifier could be useful in discussions involving more than one context.

Appositive elements are added to the context vector when the program already contains useful synonyms, descriptors, or context-specific facts that are available at run time. For example, on a data-entry screen in ACIFS, the lowest-level active object is a data field, if one is active. Associated with every data field is a data-field label (which is displayed as a prompt). If a Dynamic Help designer judged these labels to be useful as part of the message, the data-field label of the active data field could be added to the context vector as an appositive element. Appositive elements are ignored in context identification and message construction, except when a slot in the message is filled with one. Their act type or object type is undefined, and thus their level is undefined.

It should be noted that appositives would typically be determined by only one or two context-determining elements rather than by the whole context, so that a hypothetical table of contexts would not be well normalized. However, since the point is to avoid having to store such a table or retrieve from it, such relational database considerations as normalization are irrelevant. What appositives could accomplish is to bring in message-filling material from the running program, where available, thus avoiding storing that material in the Dynamic Help knowledge base.

The program that is being documented by Dynamic Help must perform *context monitoring*, so that the Dynamic Help system can perform context identification and message construction. The foregoing context model places very limited demands on the program's context monitoring capabilities. First consider the case where all the acts and objects defined for a Dynamic Help system already exist as entities in the program. For example, ACIFS has screens, documents, soldiers, inventory items, and data fields as objects that it deals with, and it has a functional organization that "sees" the acts of data

entry as parts of higher-level acts such as issuing clothing to a soldier. If the Dynamic Help designer has defined the acts and objects in the context vector strictly from among these already existing in the program, the only context-monitoring issue is their availability. At run time the Dynamic Help module must be able to retrieve each element. If the element is one that is stored in a global variable, or one that is stored in a local variable that is available while the current subroutine is suspended for Dynamic Help, or if there is a callable routine that can return the element's value, then context monitoring is already accomplished. Otherwise, it may be necessary to make some adjustments in the program to render the element's correct value accessible to the Dynamic Help system.

The other case is that an act or object needed for Dynamic Help does not already explicitly exist in the program. As an example of such a case, suppose the *active screen* identifier is needed because the message is to include an explanation of screen layout. (As will be seen below, these explanations would be stored in a Dynamic Help dictionary whose entry key would be the screen identifier.) Now suppose that the program suffers from in-line coding; for example, the programmer may have written code by which the screen is erased, new material and a prompt are displayed, user input is collected, and the screen is again erased for the next context. In such a case there would have been no current-screen object defined, and it would be necessary to sprinkle context monitoring code throughout the program. Since this would constitute an authoring task at every context, much of the automation benefit would be lost.

3.2.5 Acts, Objects, and Navigation in ACIFS Contexts

In a DBMS-based program such as ACIFS, an interactive session can be viewed as a series of database transactions that the user controls. The user exercises control by navigating via menus and cursor keys to a desired *context* where the system is ready to perform a given transaction. By invoking Dynamic Help while in a context, the user can

verify that he is in the right "place" (context), and can receive information on what the transaction would mean (its consequences if completed), how to complete it or abort it, and how to get somewhere else.

A context was defined above as the set of *active objects* and *active acts*, where there can be up to one active object of each *object type* and up to one active act of each act. Object types in ACIFS are the soldier objects, the inventory item object, the menu-screen object, the date-entry screen object, etc. Act types in ACIFS are the data-entry act, the menu-selection act, the clothing-issue act, etc. Acts are commands, procedures, functions, or tasks — data tables (e.g. the soldier table), data attributes (e.g. the SSN attribute or column heading), data values (e.g. 462-54-4872, a value of SSN for a particular soldier), menu screens, data-entry screens, items in a menu, or fields in a data-entry screen.

Dynamic Help works best when it is easy to get into a context that allows help messages to be as specific or as general as the user desires. *Premature closure*, where the user cannot highlight commands and objects without invoking them, provides shortcuts for the expert user, but prevents some detailed Dynamic Help. For example, if a user gets into a menu screen and cannot type "2" without invoking menu item "2" and going into another screen, then there can be no Dynamic Help message that explains only item"2" is detail.

Lack of a neutral context provides shortcuts for the expert user by automatically activating whatever object the designer assumes is wanted upon screen entry, but prevents some higher-level Dynamic Help. For example, if a user gets into a menu screen where item "1: is automatically highlighted and there is no neutral item, then there can be no Dynamic Help message that explains only the overall group of items. ACIFS suffers from lack of a neutral context in data-entry screens. For this reason, Captain Wolven provided a "global" sentence, which was essentially a complete message for the parent context of the current context.

Navigation from context to context is an important issue in Dynamic Help. Dr. Young compares Dynamic Help to a system of signage, as contrasted to a map. To get more information about other contexts while using only the documentation provided by Dynamic Help, the user must navigate to other contexts.

If a "global" sentence is not provided in a data screen, a message essentially identical to it is available by going to the parent menu screen, and this message was in fact available to the user earlier on the way down to the data-entry screen. It can be argued (in favor of a global sentence) that novice and intermittent users may not realize how to get more information by exploring contexts; on the other hand, it can be argued (against a global sentence) that it dilutes the message for the primary act.

The primary act (recall item 4 of the proposed context model in Section 3.2.4) is of central importance in the philosophy of Dynamic Help. The message is for the primary act, not for any alternative acts that can be performed in the same context. Although an *alternatives sentence* will be provided in Section 3.3, it is believed that alternative acts should be explained only in rare instances, to avoid diluting the message. Dr. Nathaniel Borenstein's doctoral thesis makes the point:

> I begin with the assumption that the single most important criterion by which a help system should be judged is the degree to which it facilitates the accomplishment of a particular task by a user who did not previously know how to accomplish that task (Borenstein, 1985, page 9).

3.3 Improvement to the Dynamic Help Message Structure

3.3.1 Message Structure for ACIFS Dynamic Help Prototype

In "Specifications of User Requirements, and Dynamic Help System Standards for EUS Project and CIF Conversion" (Young, 1990b), Dr. Young established a Dynamic Help message structure for ACIFS consisting of the set of sentences shown in Table 3-3.

Table 3-3 Original Eight Generic Dynamic Help Sentences

1. **Global**	("Ready to" sentence for the neutral context rather than the context of the highlighted item.
2. **Ready to**	(called "context" sentence, explaining the primary act)
3. **Navigation**	(how to go back or forward from the current context)
4. **Meaning**	(meaning of the datum being entered or selected in the primary act)
5. **Domain**	(limits on the datum being entered)
6. **Choices**	(called "content sentence, giving a list of legal or illegal entries)
7. **Format**	(example or description of acceptable format for the datum being entered)
8. **Procedure**	(description of how the entry can be completed or aborted, changed before entry, or undone later; and descriptions of quirks and alternatives)

The designer was to examine every context (about 700 of them) and author each sentence, frequently letting sentences or parts of sentences be null where not needed, and making maximal use of semi-automation techniques. These techniques included using templates (one was developed for menu screens and one for data-entry screens) to make context specification and message authoring a fill-in-the-blank task, and the use of macros for fixed text fragments and for sentences, clauses, or phrases that appeared in multiple messages. Contexts were identified by listing the names of the current screen and current data object and equating their names to variable names such as *A* and *X*, so that symbols such as «*A*» and «*X*» could be used in sentences.

User testing and experience with the messages established that the global sentence should not be first, and probably should be omitted; that the "Ready to" sentence should be first; that the navigation sentence had limited utility; and that the "Choices" sentence took too much space when list of choices was available.

At about the same time, it became clear that the ACE programming environment's separation of "choices" from "help" was a good idea. Since the next version of ACIFS was to be implemented under ACE, it was decided that the "choices" sentence should state, when choices for a primary act's entry or selection were actually available, how to access them (otherwise, as before, this sentence would be null).

3.3.2 Message Structure for Barge's "Universal" Dynamic Help

Captain Walter Barge, in his paper "Universal Software Documentation via Dynamic Help" (1991) further advanced the structure of Dynamic Help messages by identifying eight generic Dynamic Help sentence types associated with the variable parts of a Dynamic Help message. These generic Dynamic Help sentences are given in Table 3-4.

Table 3-4 The Eight Generic Dynamic Help Sentence Types

1. **Context**	Ready to «accomplish a particular task».
2. **Navigation, Global**	To return to «a previous location»«perform this action». To move to «the next location»«perform this action».
3. **Navigation, Local**	(form depends on screen design; concerns management of cursors, objects and commands on the current screen)
4. **Meaning**	«definition of the current task or field».
5. **Domain**	Acceptable entries are «list of descriptors».
6. **Content**	«a list of legal entries».
7. **Format**	«an example of a correct entry».
8. **General**	For general documentation press the F1 key.

Captain Barge made two contributions to message structure. First, by documenting the first serious attempt to render sentence structures able to be filled automatically by

context and dictionary material, his report clarified the automation issues (as summarized earlier in Section 3.2.3).

Secondly, he realized that there were many parts of messages that were *context-independent* or "general" — fragments or sentences that would be repeated in many different messages. A user does not want to be told repeatedly, "To complete the entry, press «RETURN»". Such instructions are not only independent of context, but are often at the user-action level, one level below the lowest-level *act* that accomplished something and deals with application objects rather than interface objects; thus such instructions interfere with transparency of the interface. The solution adopted by Captain Barge was to store all general (context-independent) documentation in a separate on-line document and to provide a fixed sentence telling the user how to access it. It should be noted that the definition of general documentation includes standard non-context-specific help, so that this solution also provides an appropriate way for Dynamic Help and standard help to cohabit.

3.3.3 A Proposed Dynamic Help Message Structure

In accordance with the experience gained from the previous message structures, there should be no global sentence and no navigation sentences, but there should be a context-specific sentence that could play these and other roles.

For the context-specific "ad-hoc" sentence and the "alternatives" sentence to be proposed below, the context identification will be handled by defining a *partial context* as follows: for a subset **P** of elements in the context vector, let the values of the elements define a partial context. In any dictionary whose key entry is a partial context (specifically the ad-hoc sentence dictionary and the alternatives-sentence dictionary, defined in Section 3.4), the full set of context elements will constitute the collective key field. A partial context will be defined in the collective-key fields by specifying values for the elements in **P** and leaving null the key-field values for elements not in **P**. In any current context vector

there can be null elements, whose interpretation is that there is no active act or active object of the element's type, or there is no appositive. When the context vector is compared to the collective-key fields in a dictionary, a *match* is declared if there is a match for all the elements in **P**. (Thus the nulls in the context vector are meaningful, whereas the nulls in the collective-key fields of the dictionary are "wild cards".) Table 3.5 lists the sentences in the proposed message structure.

Table 3-5 The New Eight Generic Dynamic Help Sentences

1. **Ready to:**	The context sentence names the primary act and the objects it manipulates.
2. **Ad-hoc:**	The ad-hoc sentence (usually null) gives the user any advice peculiar to the specific context of partial context (hence it is not automated) concerning quirks, consequences, special input protocols, etc. that the automated sentences cannot provide.
3. **Meanings:**	The meanings sentence (possibly null) states the meaning (if available) for each relevant active act and relevant active object in the context.
4. **Choices:**	If a list of user-action choices is available (or already on display) to invoke the primary act, the choices sentence (possibly null) tells how to retrieve it (or reminds the user it is already on display).
5. **Format:**	The format sentence retrieves, if available a format for the direct argument of the primary act.
6. **Domain:**	The domain sentence retrieves, if available, the domain for the direct argument of the primary act.
7. **Alternatives:**	The alternatives sentence gives relevant alternative acts — one of the undo or abort nature, and one, if available, of the commit or closure nature.
8. **General Help:**	The general help sentence tells how to access general (context-independent) help.

3.3.3.1 Dynamic Help "Ready to" Sentence. Recall that the *context* of an interaction is thought of as "where" the interaction is (what screen, what cursor location, etc.) together with conditions of the interaction (what command is active, etc.). Every command and every object has current status, such as being "active" or not; a list of active or otherwise special-status commands and objects completely describes the context.

Besides context, there is *state*: the current values of all variables, parameters, and data. (State includes context, but informally we can speak of state as excluding the variables that express context.)

A context description in terms of cursor location, active window or screen, and highlighted objects and commands is already available to the user before Dynamic Help is accessed. The user requires, among other things, an indication of *what can or should be done* in the context. A fundamental assumption of Dynamic Help is to associate with each context a *primary* ("Ready to") *act*. The primary act is the next thing the user and system are poised to do if this context is the intended one.

An *act*, which is something the user and system do together to manipulate application objects, is described to the user in the form of an *act clause* that is assembled with other elements to form sentences in a Dynamic Help message. An act is documented with an *act clause* which has a grammatical structure specific to the act.

Standard structure of an act clause

«verb» «direct argument» «preposition» «indirect argument»

where:

- «act-id» (the key field) contains the value of the active-act program variable that identifies this act.
- «act-type» contains a value that classifies this act by its type.

- «verb» contains the English-language verb that begins the act clause for this act. The verb need not be unique (more than one act may have the same verb in its act clause). To "delete" one type of object is a different act from to "delete" another type. For some systems, the designer may find it convenient to establish and enforce verb uniqueness.
- «darg-type» contains the value of the object type that indicates the *direct-argument*, a noun phrase, that defines which argument the act directly operates on. (The direct argument of an act is the active object having one of the object types. The value identifies which type.)
- «preposition» contains the English-language preposition, possibly with modifiers. It continues the act clause for the act and introduces the indirect argument (which grammatically is the object of the preposition).
- «iarg-type» contains the value of the object type that indicates the *indirect-argument*, the noun phrase. It defines the argument that the act indirectly operates on. (The indirect argument of an act is the active object having one of the object types, and this value identifies which type.)

The act clause has the same structure regardless of the act's act type. Every act has an act type, which is one of the following:

<u>*Act Types*</u>
command
procedure
functions
task

An act has a verb that identifies it (e.g. "add"), an act type (e.g. "command"), a direct argument pointer (e.g. the active *inventory item*), a preposition (e.g. "to"), and an indirect argument pointer (e.g. the active *table*). The act clause is constructed at run time by

consulting the dictionary for the active *act*, retrieving data, and performing further retrievals (e.g. the identity of the active inventory item) until the at clause is filled.

Examples of act clauses are:

Issue clothing **to** soldier
Designate 5-ton stake truck **as** member of convoy
Update stock record

The fixed-text portion of an act clause, shown bold in the above examples, describes the act, which is a task, function, procedure, or command. An act can consist of a sequence of lower-level acts, and the lowest-level act to be documented is a *command.*

The variable portions of an act clause consist of noun phrases, each of which describes an object that is one of the *arguments* of the task, function, procedure, or command that the act clause describes. There is no specific restriction on the numbers or kinds of arguments, but there is frequently a *direct argument*, which is the object that the act directly manipulates, and an *indirect argument*, which is the object that the act indirectly manipulates of effects.

A Dynamic Help act clause itself reads as an imperative sentence whose implied subject is the user and system. Grammatically, the direct argument is usually the direct object of the act, and the indirect argument is usually the indirect object of the act or the object of a preposition in an adverbial phrase. In the above examples, *clothing* is a direct argument and *soldier* is an indirect argument; *5-ton stake truck* is a direct argument and *member of convoy* is an indirect argument.

Objects are classified into *types*, usually identified by the name of the parent object if there is one. For example, the soldier 462-54-4872 is an object of the "soldier" type. these classifications of objects are peculiar to the application's data structure. The classifications are established so that it is always true that either one object or no object of a given type is active. The *context* is a list of active objects; for example, the active *screen*

may be "supply requisitions", the active soldier may be «null», the active *inventory item* may be "hat, hot weather", and the active *datum* (cursor location) may be "required delivery date".

Noun phrases are filled with state-specific descriptors. For example, *soldier* is an object, *soldier 462544872* is the descriptor of an instance of that object (the instance being the soldier whose social security number is 462-54-4872); similarly, *hat, hot weather* is an instance of *clothing*. Depending on the state, various Dynamic Help sentences could contain versions of an act clause:

Issue clothing **to** soldier
Issue clothing **to** soldier 462544872
Issue hat, hot weather **to** soldier 462544872

The Dynamic Help designer would not be concerned directly with these variations, because they would be constructed at runtime. For instance the act clause

Issue hat, hot weather to soldier

may never occur, because the problem is designed to issue clothing only when there is an active instance of *soldier*. Nevertheless, the Dynamic Help module would be capable of producing that act clause if (by program error or by design change) it ever happened that the cursor was on this hat in the issue screen and there was no active soldier object.

3.3.3.2 Dynamic Help Ad-Hoc Sentence. The ad-hoc sentence tells the user about any particularity regarding the specific context. In this regard, the ad-hoc sentence (a sentence which is usually «null») is a context specific message. The ad-hoc sentence can be used to inform the user of quirks in the interface, special consequences of a user action, or any special input protocol. The ad-hoc sentence has the structure:

«sentence»

where:

- The «sentence » is retrieved from the ad-hoc dictionary whenever the context vector's values obey a specified partial context requirement P.

3.3.3.3 Dynamic Help Meaning Sentence. The meaning sentence tells the semantic meaning, in application environment language, of the data objects currently active; of the act-types currently active; of the menu-item currently active; or all of the above. The meaning sentence has the structure:

«verb» is to «verb meaning»
«iarg» is «iarg meaning»
«darg» is «darg meaning»

where:

- «verb» refers to the English-language verb that begins the act clause for this act. Again, the verb need not be unique.
- «verb meaning» is the English language meaning, in application environment terms, of the «verb» (if act meanings are to be included).
- «iarg» is the indirect argument of the act clause.
- «iarg meaning» is the English language meaning, in application environment terms, of the indirect argument (if iarg meanings are to be included).
- «darg» is the direct argument of the act clause.
- «darg meaning» is the English language meaning, in application environment terms, of the direct argument.

An alternative structure of the Meaning sentence, for programs where only the darg requires explanation, is

Meaning: «darg meaning»

In ACIFS the commands are very simple and the alternative structure can be used. For programs with more complexity in its commands and functions, the first structure allows meanings to be given for the primary act, any other act, the darg, and the iarg.

3.3.3.4 Dynamic Help Choices Sentence. The choices sentence gives a list of legal or illegal entries. (Because recognitions is easier than recall, good interface design provides that a user should be able to view a list when asked to make an input that essentially chooses from one.) Dynamic Help should not provide a list when the list is intentionally withheld for quality-control, security, or validation purposes.

The choices sentence has two forms:

See the displayed list of appropriate «darg» entries.
For a list of choices «user action».

3.3.3.5 Dynamic Help Format Sentence. The format sentence exemplifies or describes the acceptable format of an entry. The format dictionary will contain an entry for every object that the programmer feels needs an example or explanation.

The structure of the format sentence is:

Format: «data field» «data field format»

where:

- the format dictionary is queried for the value «data field», and if found, the value of the format field «data field format» is returned.

3.3.3.6 Dynamic Help Domain Sentence. The domain sentence gives limits on the datum being entered, such as whether it is textual or numeric, whether it has upper or lower limits or impermissible values, and the maximum or required number of characters. The domain dictionary will contain an entry for every data-entry object that the programmer feels needs an example or explanation.

The structure of the domain sentence is:

Domain: «data field» «data field domain»
or
Example: «data field» «data field domain»

where:

- the domain dictionary is queried for the value «data field», and if found, the value of the format field «data field domain» is returned.

3.3.3.7 Dynamic Help Alternatives Sentence. An alternatives sentence tells how to undo or abort the primary act; how to commit or confirm recent actions or recently entered data; or how to perform available acts other than the primary one.

The structure of the alternatives sentence is

To «verb» «darg» «prep» «iarg», «procedure»

This is very similar to the "Ready to" sentence except that the variable slots are for an alternative act rather than the primary act, the sentence begins with "To" rather than with "Ready to", and there is a procedure (how-to) clause added at the end.

The undo/abort actions typically lose data for the lower active object or lose data for the next lower active object or abort the primary act or the next higher act. In ACIFS, pressing the delete key loses all data entered on a given screen and aborts the highest level act that is performed entirely on that screen. Therefore, for ACIFS, the effort of deleting can be described as "pressing DEL loses all data entered in this screen".

3.3.3.8 Dynamic Help General Sentence. The general sentence tells the user how to access context-independent help. The general sentence provides the novice or intermittent user with the very basic instructions. For example, a new user may never have used a menu-driven system. The general sentence provides help on navigating menu-driven systems, which in itself has nothing to do with the primary act.

3.4 A Proposed Set of Dynamic Help Dictionaries

All data needed to fill message slots must come from the context vector, from a subroutine (in the running program) that returns a value, from a database query, or from the Dynamic Help dictionaries. That is, the dictionaries must carry all message data not available from the running program or database.

In the design for a given application program, the designer will choose a specific set of detailed sentence structures. These detailed structures can be specified in the form of a *sentence specification* string that contains literal text and *slot specifications*. The slot specifications determine the structure of the required dictionaries.

The dictionaries are recursive. That is, the designer can store specifications in dictionaries, so that when the system retrieves a value from a dictionary, the value is not necessarily literal text but can be a specification that requires further evaluation.

3.4.1 Slot Specifications

A slot specification is an expression that can appear as a value in a sentence specification or in a dictionary. The expression is constructed of sub-expressions of the following forms, and usually all of the forms would be used in various places in a Dynamic Help system for an application program:

- literal text
- macro
- dictionary retrieval code
- SQL query
- name of (or pointer to) a context-vector element
- subroutine call

Except for the dictionary retrieval code, every form of sub-expression evaluates to text in one step.

3.4.2 Dynamic Help OBJECT Dictionaries

Let **D** be a dictionary, and let **k** be the key entry field in the dictionary. Let **F** be the field whose value is to be retrieved. Let **S** be a specified value that a value in **k** might equal. A dictionary retrieval code has the form

D.F (k = S)

and it specifies the **F** field in dictionary **D** for the row in which the **k** field has value **S**. A dictionary has the same structure as a relational database table, and the dictionary retrieval code is equivalent to the SQL subquery

```
select F
from D
where k = S
```

We allow **k**, **F**, **S**, and even **D** to be expressions that evaluate to the intended values.

Let **CV** be the context vector. A variation of the dictionary retrieval code is

D.F (k ≈ CV)

where the symbol "≈" indicates a match between **k** and the current values in the context vector. A match exists if every element matches. An element matches if

in k it is...	and in CV it is...
actually null	anything
the text string "null"	null
the text string "not null"	not null
«A»	an expression that evaluates to «A»

where «A» is an expression that can be evaluated.

3.4.3 Dictionaries

The dictionary structures are determined by the dictionary retrieval codes that appear in slot specifications. Consider the *act clause* from Section 3.3.3.1:

«verb» «direct argument» «preposition» «indirect argument»

Such a clause appears in the "Ready to" sentence, in the Alternatives sentence, and in any sentence (such as the Ad-hoc sentence) that is specified to include detailed mention of an act.

Let the primary act, which could be of any act type, be denoted **priact**, and let **PRIDICT** denote the dictionary for that act type. Then the act clause can be expanded:

«verb» is «PRIDICT.verb (k = priact)»

«darg» is «PRIDICT.darg (k = priact)»

«prep» is «PRIDICT.prep (k = priact)»

«iarg» is «PRIDICT.iarg (k = priact)»

Let **ACT** be an act dictionary, and let its key field **k** contain one of the values that may be in a specified act field of the context vector; denote this field **CV(A)**. Then, to support act clauses, it follows that act dictionaries must have the structure

ACT (k, verb, prep, darg, iarg)

for any act type for which act clauses must be built. The Commands dictionary in Appendix 3 is an example of such an act dictionary.

Let **FIELDS** be an object dictionary for the screen-field object type. Suppose the designer has decided that the direct argument of a particular act is to be named in a sentence as the short descriptor of the active screen field, and that the identifier of the active screen field is the "screen_field" element of the context vector. Then in the **darg** field of the act dictionary the designer can write the expression

FIELDS.short_descriptor (name = screen_field).

This requires that the dictionary of screen fields have the structure

FIELDS (k, short_descriptor)

where values of **k** are values of the screen_field element of the context vector.

If the designer also provides for storing meanings of screen fields, where the meanings sentence for objects has the structure

Meaning: «OBJ.meaning (k = darg)»

(This is an appropriate structure when the direct argument of the primary act is the only object for which a meaning, when available, is to be included in the message.)

Now FIELDS is an object dictionary. If there are meanings to be supplied for many screen fields, this dictionary could contain them:

FIELDS (k, short_descriptor, meaning)

The choices of splitting or combining dictionaries should be made on an efficiency basis. If there is to be a single object dictionary **OBJ** for objects of all types, it can have the structure

OBJ (k, short_descriptor, meaning)

of if meanings are provided for only a few objects, it could be split into

OBJSHORT (k, short_descriptor)

and

OBJMEAN (k, meaning)

Similar dictionaries of choices, formats, and domains may exist for objects. If there are only a few objects that require choices, formats, or domains, then a separate dictionary would be best. A choices dictionary may look like

CHOICES (k, choice)

or if a choice existed for most all objects, then a column "choice" could be added to the OBJ dictionary

OBJ (k, short_descriptor, meaning, choice)

The decision to add a column or add a separate dictionary would be made by the designer based on the most efficient design.

A format dictionary,

FORMATS (k, format)

and a domain dictionary,

DOMAINS (k, domain)

could also be handled by either separate dictionaries or new columns in the OBJ dictionary

OBJ (k, short_descriptor, meaning, choice, format, domain)

Regardless of how the dictionary is constructed, the designer can use the expressions

CHOICES.choice (name = screen_field)

FORMATS.format (name = screen_field)

or

DOMAINS.domain (name = screen_field)

in the respective choices, formats, or domains sentence to retrieve from the dictionary. In all of these cases, values of k are values of the screen_field element of the context vector.

3.5 Application of Automation to ACIFS Dynamic Help

3.5.1 A Context Model for ACIFS

Four levels of act and five levels of object seem adequate for ACIFS Dynamic Help context modeling.

The level-1 (lowest-level, or command-type) act in ACIFS is to enter a datum, go to another screen, or answer a yes/no question. We will call the corresponding values of the command-type context-vector element "enter," "go to," and "y/n." The level-2 (or function-type) act in ACIFS is to identify or designate an application-class object such as a soldier, inventory item, or document, which is done by entering one or more data items (on menu screens, this level of act is not needed). We will call the corresponding values of the

function-type context-vector element "id soldier," "id inv item," etc. The level-3 (or procedure-type) act in ACIFS is to perform a major process: perform the Turn In Process, Prepare a Statement of Charges, Prepare a Report of Survey, Create Supply Requisitions, Perform a Complete Issue, etc. (On menu screens, this level of act is not needed.) We will call the corresponding values of the procedure-type context-vector element "m81-Turn Ins," "m21-S/C," "m22-R/S," "m61-Create Supply Requisitions," "m72-Complete Issue," etc. The level-4 (or task-type) act in ACIFS is the overall task or group of major processes; for example, the task whose task-type context-vector element we will call "m2-Adjustment Transactions," includes preparing a Statement of Charges, a Report of Survey, handling Cash Collections, making Administrative Adjustments, etc. The level-4 task-type act upon initial entry to the system is given the value "m-Main Menu."

The level-1 (lowest-level) object in ACIFS is assigned the type designator "screen field." On a menu screen this is the highlighted menu item, and on a data-entry screen it is the cursor-resident data-entry field. Because of the strict hierarchical structure of the ACIFS screens and screen fields, these can be uniquely identified by a numbering system m*ijkl*, where the letter m is followed by a digit *i* that represents the choice on the main menu that branches towards or to the current context. The main menu is screen m; its menu items are m1, m2, etc; the menu screen that is invoked by choosing menu item m4 is screen m4, where menu items are m41, m42, etc.; when a data-entry screen is reached, say screen m444 (reached by choosing item 4 on the main menu, item 4 on the next, and item 4 on the next), the data-entry fields are numbered m444.1, m444.2, etc.

The data-entry fields are also identified uniquely with the database items they define. For example, the database item "aeb01t.siz" (the "siz" column of the table "aeb01t") is identified with data-entry field "m81.10" (the 10th field on screen m81). Both the screen field number and the database item identifier are maintained in the program, so

we put *both* in the context vector; let the screen field number be the level-1 object element, and let the database item identifier be an *appositive* element. The screen field number always exists; the database item identifier is null for fields on menu screens, but always exists for fields on data entry screens (when the item is not being stored in the database, a name such as "formonly.iqty" is available).

Exploitation of the program's maintaining two data-field identifiers by defining an appositive in the context vector is an example of the flexibility of the automation method. If ACIFS were less conveniently designed with respect to context modeling, at worst it would be necessary to maintain a Dynamic Help dictionary to identify the data item that goes with each field. On the other hand, if ACIFS domains and formats were more conveniently maintained in the program, we could define domains and formats as appositives as well and avoid keeping the domain dictionary and format dictionary in the knowledge base.

The other object in the context of ACIFS are "inventory item" (level 2), "soldier" (level 3), "document" (level 4), and "screen" (level 5). Although the screen is implied by the screen field number, the program's available screen names (e.g. Turn Ins, Supply Requisitions, Issue Additional Items) are sufficiently descriptive to be usable in messages; thus there is no necessity to keep screen descriptors in a dictionary.

Appendix 2 illustrates the context model for a sample of actual contexts in ACIFS. In Appendix 2, the screen field numbers for menu-screen fields and the database item identifies for data-entry fields are shown in a single column labeled "Screen Fields."

3.5.2 A Message Model for ACIFS

Applying the proposed Dynamic Help message structure of Section 3.3.3 to ACIFS, we have the following eight sentence structures:

1. **"Ready to" sentences** for ACIFS can all document the level-1 command acts (enter, go to, and respond), since in every ACIFS context the user is expected to enter a

datum, go to a different screen, or respond to a yes/no question. A typical "Ready to" sentence will be

Ready to enter social security number of soldier.

"Ready to" is fixed text. The verb "enter" is «verb» from the dictionary of commands for the primary act of "enter", which is the level-1 act for this context as read from the command element of the context vector. The noun phrase "social security number" is «short descriptor» from the dictionary of screen fields for the active screen field as read from the context vector as the **darg** (since a screen field is always active in ACIFS, the darg is always the screen field). The word "of" is «prep» from the dictionary of commands for the primary act. It introduces the **iarg**, which is the darg for the **hiact**, which in turn is the next higher level active act above the level of **priact**. (Recall **priact** is the lowest-level act in the context vector.) The phrase would have been e.g. "462544862" if the context had contained an active soldier.

2. **Ad-hoc sentences** are provided for partial contexts. An example is the contexts where the active function (element 3 of the context vector) is "id inv item". Note from Appendix 2 that many of the contexts in the sample of contexts have this partial context. For those on screen m81, (element 5 of the context vector gives the screen), a special instruction is needed and is stored in the ad-hoc dictionary. The instruction is retrieved and included whenever the context is

(,,id ivn item,,Turn Ins,,,,).

3. **Meanings sentences** for ACIFS are provided whenever the darg is a screen field for which the dictionary of screen fields provides an nun-null entry. Since the meanings were inherited from previous documentation, they are stored as complete sentences rather than as predicates. For example, the meanings sentence

The Demand Code indicates whether the demand is recurring or nonrecurring

is stored whole because it came that way. If these were being authored by the message designer, the equivalent meanings sentence would be

Meaning of demand code: indicates whether the demand is recurring or nonrecurring

where "Meaning of" and ":" are fixed, "demand code" is the short descriptor, and the remainder is kept in the meaning field of the screen fields dictionary.

4. **Choices sentences** are provided in ACIFS whenever a displayed list is on view or can be retrieved. When the darg is the screen field for the datum "aeb15t.dic", for example, the choices sentence (stored whole and kept in the choices dictionary under this database item identifier is

See the displayed list of appropriate DIC entries.

5. **Format sentences** are stored whole in the dictionary of formats under the database item identifier and are retrieved whenever the darg's identifier is listed in this dictionary.

6. **Domain sentences** are stored whole in the dictionary of domains under the database item identifier and are retrieved whenever the darg's identifier is listed in this dictionary. A short example, when the darg is "aeb02t.sex", is

Acceptable entries are M for Male and F for Female.

Domain sentences are inherited from existing ACIFS documentation. If they had been individually authored, the above sentence would have appeared as

Domain: M, F (male, female)

where "Domain:" is fixed text.

It should be noted that both formats and domains could have been finessed in ACIFS by putting the available format and domain specifications in the context vector.

That would be unwise for two reasons: they are available in cryptic codes that themselves would require explanation, and worse, giving formats and domains in every message would dilute the message. The dictionaries allow the designer to provide format and domain sentences only where needed, as for hat sizes, while avoiding unnecessarily telling the user that no more than 99999 hats can be ordered in a single requisition.

7. **Alternatives sentences** are unnecessary in ACIFS, because every context is associated with a single expected user action, except that using the ESC key to commit data, using the DEL key to abort data entry, and using the BACKSPACE key and over-striking to make corrections before entry are all available everywhere and are documented in the general documentation to which the General Sentence refers. Note that the ad-hoc sentences provide a less structured way of documenting alterative acts. An alternatives sentence, which is structured like a "Ready to" sentence with an added procedure (how to) clause, should be used only when the primary act is not the only "usual" thing to do in the context.

8. The **General Help sentence** for ACIFS is the fixed text

To see general documentation, press F1.

A dictionary of general sentences (see Appendix 3) should be prepared for ACIFS in order to separate the general help for all menu screens from that for all data entry screens. When the user accesses general documentation, only the general documentation pertinent to the current kind of screen is retrieved.

3.5.3 ACIFS Dictionaries

For ACIFS, where all contexts have an active command, and thus all "Ready to" sentences refer to a level-1 act, it is convenient to let acts of each level have separate dictionaries. Let these dictionaries be

COMMANDS (command, verb, prep, darg, iarg) *level 1*

FUNCTIONS (function, verb, darg) *level 2*

PROCEDURES (procedure, screen) *level 3*

TASKS (task, screen) *level 4*

The key (underlined) fields of the respective act dictionaries hold the values from the respective act fields of the context vector. In act dictionaries that have a **verb** field or a **prep** field, their contents are text provided by the designer to fill sentence slots. The **darg** field identifies the type of active object that is the direct argument of the act; to make the dictionary readable, let this identification be by the name of the object type rather than by a pointer to the context vector. It can also contain literal text in quotation marks, a data base item identifier, of both, separated by the word «or», in which case the value of the database item is used if not null, otherwise the literal text is used. Thus a darg sentence slot can be filled with a context vector value, a literal text string, or a database value, depending on what is found in the dictionary and database. The **iarg** field (which occurs only in the command dictionary) contains a dictionary retrieval code identifying how to determine which object is the indirect argument of the act. For ACIFS, only the **enter** command has an iarg, and its code is

FUNCTIONS.darg (function = CV(function))

Thus the iarg is the value from the **darg** field of the **FUNCTIONS** dictionary for the function that is the active function as read from the function (level-2 act) element of the context vector.

The **PROCEDURES** and **TASKS** dictionaries are not used in messages for ACIFS, since these higher-level acts are simply names for related groups of lower-level ones.

The act dictionaries for ACIFS are illustrated in Appendix 3, where for the sake of completeness the procedures and tasks are given dictionaries that store their names and screens.

ACIFS requires only one object dictionary, the one for screen fields.

FIELDS (name, short_descriptor, meaning)

Because of the relationship in ACIFS whereby every data-entry field is associated with a database item, the item identifies the value in the key field; for menu screens, the screen field number is used. These values come from the context vector. See Appendix 3 for a sample of the dictionary of screen fields for ACIFS.

Other ACIFS objects (inventory items, soldier, document, screen) require no dictionary, because their attributes necessary to fill sentence slots are available either from the context vector or the database.

There are three dictionaries that give additional information about the **darg**; these are the dictionary of choices, the dictionary of formats, and the dictionary of domains:

CHOICES (name, choice)

FORMATS (name, format)

DOMAINS (name, domain)

For each of these dictionaries the key (name) field is the database item identifier, and the other field is text, macros, dictionary retrieval codes, and SQL queries. When any macros, codes, and queries are evaluated and assembled with the text, the resulting text string is ready for filling the sentence slot in the Choices sentence, Format sentence, or Domain sentence, respectively. These sentences are null unless a dictionary has a row for the current darg.

The ad-hoc dictionary has the structure

ADHOC (context, sentence)

where the key field is a partial context. Given the context model for ACIFS (Section 3.5.1), the illustrative partial context

(,,id soldier,,Turn Ins,,,,)

can be the value of **ADHOC.context**. When it is, the sentence **ADHOC.sentence** in the same row of the ad-hoc dictionary will be assembled into the messages if the actual context has the value "id soldier" as its third element (the level-2 act element) and the value "Turn Ins" as its fifth element (the level-5 object element).

Finally the General Help dictionary contains fixed text for each of two partial contexts — one for contexts in which the active screen field is a menu item and one for contexts in which the active screen field is a data-entry field. Let there be a subroutine GENCON that returns either the text value "menu-items" or the text value "data-items," determined by inspecting the context vector. The dictionary is

GENHLP (context, message)

The General Help message is filled with the text GENHLP.message for the row where GENHLP.context equals the value returned by GENCON. (Recall this is the message retrieved by invoking General Help; the General Help sentence in the General Help message is fixed-text sentence that says how to invoke the General Help message.)

An experimental test of the automation proposed in this chapter is reported in Chapter 4.

CHAPTER IV

TESTS OF AUTOMATION VALIDITY AND EFFICIENCY

In Chapter III a method of automating Dynamic Help was developed and illustrated. This chapter reports an experiment to evaluate that method. The experiment compares non-automated, semi-automated, and automated Dynamic Help with respect to cost and quality.

4.1 Experimental Objectives

4.1.1 Treatments: Non-automated, Semi-automated, Automated

Recall that a Dynamic Help message depends both on the context and the data state. There is one message for each context, and the message can contain state-dependent values that are determined when the message is invoked.

A *non-automated* Dynamic Help message is one that is authored. The designer provides text for all sentences of every message. The text can include macros, assumed to be 5 characters long, which can retrieve current values of variables up to 160 characters long; macros can also retrieve fixed text fragments. At run time, the message is retrieved and its macros are filled. The *message table* and *macros table* constitute the knowledge base for a non-automated Dynamic Help module. Many of the benefits of automation can be obtained without resorting to the method proposed in Chapter III.

A *semi-automated* Dynamic Help message is one that takes advantage of cost-effective ad-hoc automation opportunities discovered by the designer. For example, a designer may find that the messages for a class of contexts are so similar that it is more

efficient to store the variations from a standard than to store all the whole messages. Templates for sentences or for whole messages can be defined.

It is suspected that many application programs have special structures that can be exploited, and that (1) a designer is likely to take advantage of some special structures rather than blindly follow the non-automated procedure, and (2) some special structures are likely not to be exploited by the proposed automation method (because of its generality). For these reasons, it would not be valid to compare automated Dynamic Help only with non-automated Dynamic Help. Semi-automated Dynamic Help must be a candidate.

It is impossible to define, in general, a semi-automated Dynamic Help design. However, for purposes of the present experiment, the existing Dynamic Help system in the ACIFS program, which generated the messages in Captain Wolven's user tests, is semi-automated. Separate templates were prepared for messages for menu screens and data-entry screens, certain closely-related contexts were grouped, macros were not confined strictly to retrieval from memory but were allowed to be procedures that could compute a result or query it from the database, and it was felt that some degree of automation was being achieved. We take an operational viewpoint and define the ACIFS prototype messages as representative of semi-automated Dynamic Help.

An *automated* Dynamic Help message is one generated according to the method proposed in Chapter III. Its knowledge base consists mainly of dictionaries of acts, objects, and relationships, not a message table. For purposes of this experiment, the tradeoff between context-generation logic and relationship dictionaries is resolved not for overall efficiency but to avoid having the automated system place extra demands on context generation. That is, no attempt will be made to modify the program so that the context vector would carry appositives that could directly fill sentence slots; otherwise it would be necessary to include context monitoring as a measured cost element.

4.1.2 Hypothesis: Lower Cost for Comparable Quality

It is hypothesized that automated Dynamic Help can provide messages at lower cost than semi-automated or non-automated Dynamic Help, when the messages are of comparable quality. This hypothesis subsumes the more basic one that Dynamic Help automation is valid in practice, which will be considered established if samples of automated messages indeed are of comparable quality.

We do not treat this as a statistical hypothesis, and a positive result will not support a strong conclusion. We will compare non-automated, semi-automated, and automated messages for ACIFS, which is a representative Army DBMS-based program, but one that cannot necessarily be claimed to be typical of any defined population of such programs. A positive result will allow only the conclusion that automation is successful for ACIFS which in turn will support a claim that Dynamic Help automation holds promise.

4.2 Experimental Design

4.2.1 Cost Measures

It is believed that automation should have several cost-reducing effects: it should reduce authoring effort, reduce the required size of the Dynamic Help knowledge base, and make the Help system able to be easily updated when the software is revised. On the other hand, it is believed that automation will add to the required run-time processing.

Size of the knowledge base will be used as an indicator of both authoring effort and memory requirement. The Dynamic Help prototype knowledge base for ACIFS has a directly measurable size in bytes (characters). The equivalent non-automated knowledge base has a size that can be estimated on the basis of removing the effects of semi-automation. The size of the automated knowledge base is directly measurable if the dictionaries are actually constructed for a sample of messages.

Other cost measures will be discussed informally, but are assumed to be less important than size of the knowledge base in indicating cost.

4.2.2 Quality Measures

It is believed that automation can either reduce or increase overall message clarity: the ability of an author to adjust clarity of each message by rewording or adding material should make semi-automated or non-automated messages clearer than automated ones; on the other hand, the presumed brevity and consistency of automated messages could perhaps make them clearer. With respect to clarity, we take a conservative stance: if the existing semi-automated message actually imparts specific information that the corresponding automated message omits, we will arbitrarily add clauses or phrases to the ad-hoc sentence of other appropriate sentence of the automated message, until it can be claimed that the automated message is at least as clear as its counterpart. This, of course, will add to the size of the knowledge base, so that any potential clarity decrement is translated to a cost increase.

Because the aim of semi-automation is to produce essentially the same messages as the non-automated messages, we assume the non-automated messages would be identical to the actual ACIFS messages.

4.2.3 Method of Comparison

A set of user tasks will be taken from the user-test project administered by Captain Wolven (Wolven, 1991). This set of user tasks (listed in the following section) gives rise to 46 contexts. In this group of contexts, Captain Wolven found that several of the Dynamic Help messages were relatively frequently accessed and were helpful.

For the same group of contexts, automated Dynamic Help messages will be generated by applying the proposed method of Chapter III. These will be compared, message by message, with the corresponding messages user-tested by Captain Wolven (see

Tables 4-1 and 4-2 in the following sections). The automated messages will each be augmented to match clarity levels (see Table 4-4 in Section 4.3.4), producing a set of quality-adjusted automated messages.

The rows of the message table and the macros table actually accessed by the semi-automated "Wolven" messages will be extracted, and the number of characters will be counted. (These messages can be called "Wolven" messages for convenience, and it is appropriate since she not only user-tested them, but previously authored them.) Call the results N_s, the size of the Dynamic Help knowledge base for the semi-automated sample.

N_n, the size of the Dynamic Help knowledge base for the corresponding hypothetical non-automated sample will be estimated by backing out the semi-automation techniques (where messages for more than one context were combined, macros will be defined where they are more efficient than duplicate section of messages).

N_a, the size of the Dynamic Help knowledge base for the automated sample, will be estimated as the size of the Dynamic Help dictionaries. This includes all relationship dictionaries, even if they restate features of the program's data structure that could alternatively be extracted by altering the context-generation logic.

Because the design of the automated Dynamic Help system is adjusted to equalize the context-generation function for all three treatments, the code size for context generation is not included in N_s, N_n, and N_a. (Context-generation code for ACIFS occupies about 5M + N lines of code, where M is the number of screens and N is the number of data fields.)

4.3 Message Samples Generation

4.3.1 User Task Sample

After a semi-automated prototype Dynamic Help system had been added to ACIFS, the user test designed and administered by Captain Wolven was based on user tasks that

were intended to represent the most challenging tasks (because users would not access Dynamic Help during non-challenging tasks) among the common tasks to be performed by warehouse clerks using ACIFS. Among the tasks for which users frequently accessed Dynamic Help and found the messages helpful were those described by Captain Wolven as follows (Wolven, 1991):

1. Task 11, a single task under the Clothing Turn In Process Menu, consists of posting of a manual turn-in of a soldier's clothing and equipment.
2. Task 12, a sequence of two subtasks under the Adjustment Transactions Menu, consists of:
    - 12.2 Posting a Statement of Charges (S/C)
    - 13.3 Modifying a Report of Survey (R/S)
3. Task 13, a single task under the Document Register Actions Menu, consists of:
    - 13.1 Creating four supply requisitions
4. Task 14, a sequence of two subtasks under the Document Register Actions Menu, consists of:
    - 14.1 Posting a receipt
    - 14.2 Correcting an incorrect receipt posting.
5. Task 15, a sequence of two subtasks under the Clothing Issues Process Menu, consists of:
    - 15.1 Posting a completed clothing issue
    - 15.2 Posting an issue of due-out clothing items

4.3.2 Wolven Message Sample

The user tasks listed above carry the user through 46 contexts for which Dynamic Help messages were available. These semi-automated messages constitute the sample for the present experiment.

Table 4-1 illustrates 6 of the 46 messages from Captain Wolven's semi-automated Dynamic Help. A full listing of all Captain Wolven's semi-automated Dynamic Help messages in the sample is given in the left column of Appendix 1.

Table 4-1 Illustration of Six Semi-Automated Dynamic Help Messages

Message from Context 1 **CLOTHING TURN IN PROCESS** Ready to perform a complete or partial turn in, print Clothing Record (DA 3645), or perform a direct exchange of sized items for an individual who has completed initial issue. To go to Clothing Turn In Process, press Return Key.	**Message from Context 7** **CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key.
Message from Context 2 **CLOTHING TURN IN PROCESS** Ready to perform a complete or partial turn in, print Clothing Record (DA 3645), or perform a direct exchange of sized items for an individual who has completed initial issue. **Turn In Process** Ready to process a turn in of issued items. Do not use this option if an individual has not completed the issue cycle or had completed a turn in already.	**Message from Context 8** **Stock Number** Ready to enter the Stock Number of a new requisition. To go to Document Identifier Code, enter the Stock Number. Acceptable entries are zero and positive numbers. Example: 8415011841352 Note: Filling the field signals completion, and the cursor jumps to the Document Identifier Code field. To make a correction while still in the Stock Number field, use BACKSPACE and over-strike incorrect data.
Message from Context 3 **TURN INS** Ready to process turn in of issued items for an individual. To return to CIF Clothing Turn Ins Menu, press DEL (Delete) Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **SSN** Ready to enter the individual's SSN. Examples: 123456789 for U.S. Soldier K23456789 for KATUSA KC3456789 for Korean Civilian To finish entering SSN, press Return Key.	**Message from Context 9** **Document Identifier Code** Ready to enter the Document Identifier Code (DIC) of the supply requisition. NOTE: This entry is required to process the transaction. To go to Quantity Ordered, enter the Document Identifier Code. The DIC consists of 3 letters/digits which identify the requisition type. LIST: A01 - Overseas Shipment with NSN A0A - Domestic Shipment with NSN A05 - Overseas Shipment without NSN A0E - Domestic Shipment without NSN Note: Filling the field signals completion, and cursor jumps to Quantity Ordered field. To make a correction while still in the Document Identifier Code field, use BACKSPACE and overstrike incorrect data.

The message sample for non-automated messages is taken as equal to the sample of semi-automated messages, since there is no basis for assuming how the ad-hoc semi-automation techniques applied by Captain Wolven might have changed the messages from what they otherwise would have been.

4.3.3 Automated Message Sample

Each Wolven message comes from a specific context. In the automated message scheme, 12 of these contexts, including he ones that give the six messages illustrated in Table 4-1, are as shown in Table 4-2a and Table 4-2b. All 46 sample contexts are listed in Appendix 2.

Table 4-2a Act Fields for Twelve Contexts

	Contexts - Acts			
Context Key	4-Task	3-Procedure	2-Function	1-Command
context 1	m-Main Menu			go to
context 2	m8-Clothing Turn In Process Menu			go to
context 3	"ditto"	m81-Turn Ins	id soldier	enter
context 4	"ditto"	m81-Turn Ins	id soldier	enter
context 5	"ditto"	m81-Turn Ins	id inv item	enter
context 6	m-Main Menu			go to
context 7	m6-Document Register Actions			go to
context 8	"ditto"	m61-Create Supply Requisitions	id inv item	enter
context 9	"ditto"	"ditto"	id inv item	enter
context 10	"ditto"	"ditto"	id inv item	enter
context 11	"ditto"	m72-Complete Issue	id soldier	enter
context 12	"ditto"	m76-Issue Additional Items	id inv item	enter

Table 4-2b Object Fields for Twelve Contexts

Context Key	Contexts - Screen	Objects: Document	Soldier	Inventory Item	Screen Field
context 1	Main Menu				menu item 8
context 2	Clothing Turn Ins				menu item 1
context 3	Turn Ins				aeb02t.ssn
context 4	Turn Ins		SSN		aeb02t.uic
context 5	Turn Ins		SSN	LIN	formonly.iqty
context 6	Main Menu				menu item 6
context 7	Document Register Actions				menu item 1
context 8	Supply Requisitions				aeb15t.nsn
context 9	Supply Requisitions			NSN	aeb15t.dic
context 10	Supply Requisitions			NSN	aeb15t.qty
context 11	Complete Issue				aeb02t.ssn
context 12	Complete Issue				aeb02t.ssn

The 12 contexts in Table 4-2a and Table 4-2b were generated by the context-generation method given in Section 3.5.1. Actual generation of the contexts was done by hand following the method, since context-generation logic implementing the method has not yet been installed in any version of ACIFS.

Each of the 46 sample contexts was used as the input for message construction by the method given in Section 3.5.2. Actual generation of the messages was done by hand following the method, since this method's message construction logic has not yet been implemented into a Dynamic Help module for ACIFS. Message generation was done in two steps: first, the Dynamic Help dictionaries given in Section 3.5.3 were filled with entries for this message sample only, omitting every row and column that would never be accessed in constructing any message in the sample. These partial dictionaries are listed in Appendix 3.

Finally, the partial dictionaries were accessed to produce the sample of automated messages. Table 4-3 illustrates the sample of automated messages. Note that they differ

from the Wolven messages illustrated in Table 4-1. A full listing of all automated messages in the sample is given in the right column of Appendix 1.

Table 4-3 Illustration of Automated Message Sample

Message from Context 1 **CLOTHING TURN IN PROCESS** Ready to go to clothing Turn In Process Menu. Clothing turn in processes include: a complete or partial turn in, print clothing record, or perform a direct exchange of sized items for an individual who has completed initial issue. To see general documentation, press F1.	**Message from Context 7** **Create Supply Requisitions** Ready to go to create supply requisitions. The system will assign the document number. Once the NSN, document identifier code, quantity ordered, and unit of issue entries are completed and no other change is desired, press the ESC key. To see general documentation, press F1.
Message from Context 2 **Turn In Process** Ready to go to Turn In Process. The turn in process will process a turn in of issued items. Do not use this option if an individual has not completed the issue cycle or had completed a turn in already. To see general documentation, press F1.	**Message from Context 8** **Stock Number** Ready to enter national stock number of inventory item. Acceptable entries are zero and positive numbers. Example: 8415011841352 To see general documentation, press F1.
Message from Context 3 **SSN** Ready to enter the social security number of a soldier. The turn in process will process a turn in of issued items. Do not use this option if an individual has not completed the issue cycle or had completed a turn in already. Examples: 123456789 for U.S. Soldier K23456789 for KATUSA KC3456789 for Korean Civilian To see general documentation, press F1.	**Message from Context 9** **Document Identifier Code** Ready to enter document identifier code of «LIN.description». NOTE: This entry is required to process the transaction. The DIC consists of 3 letters/digits which identify the requisition type. Examples: A01 - Overseas Shipment with NSN A0A - Domestic Shipment with NSN A05 - Overseas Shipment without NSN A0E - Domestic Shipment without NSN To see general documentation, press F1.

4.3.4 Quality Adjusted Automated Message Sample

In some sentences, the Wolven message sample could impart specific information that the corresponding automated message omitted. These instances were identified, and the automated messages were augmented to impart the same information. The Ad-Hoc Dictionary in Appendix 3 contains these augmentations.

4.3.5 Differences in Semi-Automated and Automated Messages

The automated messages in Table 4-3 do not contain the "global" reference that the semi-automated messages in Table 4-1 contain. The author decided to focus the automated Dynamic Help messages on the primary act. While this resulted in a smaller automated message, it did not affect the size of the dictionaries necessary to construct the automated messages. The global portion of the semi-automated messages is included in the dictionaries for the automated messages. For example, in the message from context 2 in Table 4-1, the global portion of the message is

> **CLOTHING TURN IN PROCESS**
> Ready to perform a complete or partial turn in, print Clothing Record (DA 3645), or perform a direct exchange of sized items for an individual who has completed initial issue.

This text is stored in the automated message dictionaries, although it does not appear in the automated message from context 2 in Table 4-3. It is included as the primary act of context 1 in Table 4-3. To add the global portion of the semi-automated messages to the automated messages would not require adding data to the automated message dictionaries.

Likewise, the general documentation of the semi-automated messages does not appear in the automated messages. General documentation is reference in all automated messages by the fixed text phrase

To see general documentation, press F1.

The specifics of the automated message general documentation is included in the GENHLP dictionary (see Appendix 3).

The automated messages of Table 4-3 are all readable, but would benefit from the insertion of definite and indefinite articles. All 46 sample message pairs (semi-automated and automated) can be examined side-by-side in Appendix 1.

It is believed that the automated messages are similar in quality to the semi-automated messages and the usability results obtained by Captain Wolven are applicable to the automated messages.

4.4 Cost Analysis

The primary indicators of cost are N_s, N_n, and N_a as defined in Section 4.2.3.

The semi-automated message sample listed in Appendix 1 contains 15,844 characters, counting each macro call as 5 characters, and the corresponding rows of the macros dictionary contain 4437 characters, for a total of $N_s = 15,844 + 4437 = 20,281$.

It turns out that no semi-automation techniques that affect knowledge storage were used in this sample, so $N_n = N_s$.

For each of the 46 semi-automated Wolven messages, several counts were made and recorded in Table 4-4:

Count 1. Gross size, the number of characters as printed,

Count 2. Global sentence quantity, the number of characters in global sentences,

Count 3. Global sentence size, the size of global sentences,

Count 4. Local repeated sentence quantity, the number of local repeated sentences,

Count 5. Local repeated sentence size, the size of local repeated sentences,

Count 6. Macro quantity, the number of macros,

Count 7. Macro size, the number of characters in macros,

Count 8. Stored Size. Adjusted size of each message.

Table 4-4 Size Computations of Semi-Automated Wolven Messages

context key		Global sentence		Local Repeated Sentence		Macros (larger than 35 chars)		Stored Size
	Gross size	Qty	Size	Qty	Size	Qty	Size	
1	251	1	199					57
2	368	1	199					174
3	409			1	170			244
4	793	1	631					167
5	786	1	631					160
6	834	1	631					208
7	1080	1	631					454
8	935	1	631					309
9	873	1	631					247
10	968	1	631			1	70	277
11	785	1	591					199
12	741	1	591					155
13	755	1	591					169
14	532							532
15	904					1	198	711
16	889	1	549					345
17	793	1	549					249
18	831	1	549					287
19	888					1	198	695
20	699	1	364					340
21	605	1	364					246
22	642	1	364					283
23	621					1	198	428
24	636					1	198	443
25	1064	1	660			1	45	369
26	1306	1	660			1	45	611
27	1034	1	660			1	45	339
28	1227	1	660			1	45	532
29	1230	1	660			1	45	535
30	1217	1	660			1	45	522
31	1148	1	660			1	45	453
32	1099	1	660			1	45	404
33	1209	1	660			1	45	514
34	1247	1	660			1	45	552
35	1250	1	660					595
36	1193	1	660			1	45	498
37	1051	1	660			1	45	356
38	513					1	198	320
39	622					1	198	429
40	727	1	557	1	170			10
41	852	1	557					300
42	801	1	557					249
43	570					1	198	377
44	577	1	403	1	170			14
45	647	1	403					249
46	635	1	403					237
Totals:	38837		20787		510		1996	15844

The number of characters as printed (count 1) includes blanks and is the number of characters that would have been authored and stored if no semi-automation had been performed.

The Wolven messages contain a global sentence applying to the whole screen whenever the context is that of a data entry on a data entry screen. Count 2 is the number of these in the message (zero or one), and Count 3 is the number of characters. Each global sentence is assumed to be stored in a global sentence dictionary, once, and called for each message.

The Wolven messages sometimes contain a repeated local sentence telling how to enter a datum that is entered on more than one screen (SSN, Total Price, and NSN). Count 4 is the number of these in the message (zero or one), and Count 5 is the number of characters. Each local repeated sentence is assumed to be stored in a local-repeated-sentence dictionary, once, and called for each message.

The Wolven messages sometimes contain stock phrases of 35 characters or more that appear in at least two messages and are not already semi-automated by occurring only as parts of global sentences or local repeated sentences. Count 6 is the number of these (zero, one, or more), and Count 7 is their total number of characters. Each stock phrase of 35 characters or more that appears in at least two messages is assumed to be stored in a macros dictionary, once, and called for each message.

The number of characters actually stored for a Wolven message, exclusive of dictionary overhead, is Count 1, less the size of all called material (the sum of Counts 3, 5, and 7), plus the size of all calls (the sum of 5 characters per call times the sum of Counts 2, 4, and 6). This result is recorded in Table 4-4 as Count 8. The sum of Count 8 is **15,844**. This is the total number of characters stored, exclusive of overhead, for the 46 semi-automated Wolven messages.

The dictionary overhead for the semi-automated Wolven messages is the total of the sizes of distinct global sentences, local repeated sentences, and macros. Table 4-5 summarizes the dictionary overhead.

Table 4-5 Dictionary Overhead for Semi-Automated Wolven Messages

Global Sentences:	3954 characters
Turn-In Process	199 characters
Manual Turn-In	631 characters
Manual Turn-In Items	591 characters
Statement of Charges	549 characters
Report of Survey	364 characters
Supply Requisitions	660 characters
Complete Initial Issue	557 characters
Issue Additional Items	403 characters
Local Repeated Sentences:	170 characters
SSN	170 characters
Macros:	313 characters
Enter Y and press Return Key	70 characters
Use BACKSPACE and	45 characters
To make another selection	198 characters
TOTAL DICTIONARY OVERHEAD	**4437 characters**

N_S is the sum of characters stored plus the dictionary overhead:

$$N_S = 15{,}844 + 4437 = \mathbf{20{,}281} \text{ characters.}$$

To provide the 46 semi-automated Wolven messages, one would have to author and store this number of characters.

The 46 automated messages are constructed entirely from dictionaries, context structure, and sentence structures. The automated messages themselves are listed in Appendix 1, right column, including all the ad-hoc sentences needed to duplicate the ad-hoc material in the Wolven messages, but excluding the global sentences in the Wolven messages.

Recall that the global sentences constitute, at most, the entire Dynamic Help message for the parent context, which for a data entry screen is the context for being on the screen with no active data-entry field. This context (hypothetical for ACIFS, since the topmost data-entry field is activated upon screen entry) is almost identical to the context in the parent menu screen where this screen's name is highlighted. As a consequence, we observe that all dictionary entries needed to support the global sentences already exist in the sample dictionaries.

Appendix 3 contains all the dictionaries, filled with the entries necessary to fill the 46 sample messages, including quality augmentation. Several rows and columns are never accessed by the 46 sample messages, but they are included as necessary to the structure of the dictionaries. The number of characters in these dictionaries, for purposes of this experiment, includes all column headings and key fields, because these must all be authored and stored. The number of characters in the Dynamic Help dictionaries for the automated message sample is 7939, which was determined by counting the number of characters in Appendix 3 including column headings and all fields, with each null entry counted as one character.

The context structure (although not all of the context fields are accessed in the sample of automated messages) contains 9 attribute type names (see appendix 2), which could reasonably be allocated 20 characters apiece; for a total of 180 characters of context fields; a data structure for assigning each act to an act type and each object to an object type, which is assumed to occupy 200 characters; and a data structure for assigning a level to each type, which is assumed to occupy 5 characters per type, or 45 characters. The total number of characters of context structure needed for the 46 automated messages is:

$$180 + 200 + 45 = 425 \text{ characters}$$

The sentence structures contain fixed text and dictionary calls, as listed in Table 4- . Each call is assumed to occupy t characters. The total number of characters in sentence structures is 160.

The total number of characters to be authored and stored for automated Dynamic Help for the 46 sample contexts is thus:

$$N_a = 7{,}939 + 425 + 160 = \mathbf{8{,}524}$$

Table 4-6 Sentence Structure Size for Automated Messages

Sentence	Number of characters of fixed text	Number of calls	Total number of characters
1. Ready to	8	4	28
2. Ad-hoc	0	1	5
3. Meaning	0	1	5
4. Choices	38	1	43
5. Format	7	2	17
6. Domain	8	2	18
7. Alternatives	0	1	5
8. General Help	39	0	39
			160

The ratio $N_s/N_a = (20{,}281)/8{,}524 = 2.38$ is a measure of automation efficiency for the sample. Automation reduces the required amount of authoring and storing by the factor N_s/N_a.

Automation Efficiency for a Complete Dynamic Help System

The ration N_s/N_a from the sample of 46 contexts underestimates the automation benefits that would be realized in a complete program. The knowledge base for automated Dynamic Help is already adequate, or almost adequate, for many more than the 46 contexts

in the sample, whereas the message file for semi-automated Dynamic Help would grow almost linearly with the number of contexts.

The total number of reachable contexts in ACIFS is about 700. From Table 4-4 for the semi-automated sample, we can see that the stored size is about 40% of the gross size. Since the total dictionary overhead (Table 4-5) is only about 12% of the gross size, the overhead is not a major factor. Let us make the conservative assumption that the overhead will not grow significantly as the number of messages increases from 46 to 700 On this basis, the estimated stored size for 700 semi-automated messages is the 46-message overhead plus 700/46 times the 46-message stored size:

$$4437 + (700/46)\ (15{,}844) = 245{,}541 \text{ characters}$$

The automated sample has dictionary entries for 4 task-type acts out of the 9 available in ACIFS; 6 procedure-type acts out of the 100 available (6 screens of 100 available were encountered in the sample); 29 data-entry items out of an estimated total of 122 available (122 fields throughout ACIFS where the user can make a unique data entry not made elsewhere); 17 domain sentences out of an estimated 70 items throughout ACIFS requiring domain sentences; no format sentences, whereas all of ACIFS would require about 15; 2 choices sentences, whereas all of ACIFS would require about 5; and 4 ad-hoc sentences in 46 contexts.

From these facts and estimates, we can estimate that the act dictionaries for all of ACIFS would increase in size by a factor of about 15, which would be a linear increase (the 46 sample contexts are about 1/15 of the total number of contexts, and the acts encountered are about 1/15 of the total number of acts); but the object dictionaries would increase only by a factor of about 4.2. Since the object dictionaries contain the bulk of the stored characters in dictionaries, it would be conservative to estimate that the total

dictionary size would increase by a factor of 5. The context structures and sentence structures would remain the same sizes as for the sample.

In summary, a rough estimate of the number of characters required for the automated Dynamic Help knowledge base for all of ACIFS is

$$5\ N_a = 42{,}620 \text{ characters}$$

The ratio of 246,505 to 42,620 is about 5.8. For a complete software package such as ACIFS, automation of Dynamic Help could be expected to require about 1/5 the authorship effort and 1/5 the memory space compared to semi-automated Dynamic Help.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

This chapter examines the automatability and flexibility of Dynamic Help and the generality of the context model and message structure (does the automation process apply to other DBMS-based programs, and to non-DBMS-based programs?) and examines the cost (in terms of required designer's effort) and applicability of the automation process to various kinds of interactive programs. A brief discussion of requirements for a Dynamic Help Generator — a designer's tool that would aid in the construction of Dynamic Help dictionaries — is given. Development of this Generator and a Dynamic Tutorial system is recommended

5.1 Conclusions

For one DBMS-based US Army installation-level software product, ACIFS, it has been shown that it is feasible to prepare a set of Dynamic Help dictionaries that allows a Dynamic Help module automatically to identify an interaction context and automatically to construct a clear and complete Dynamic Help message for each context at run time.

As was shown in Chapters III and IV, Dynamic Help can be automated. By applying the developers tools of Figure 5-1, automated Dynamic Help was added to an existing U.S. Army Installation Software program. The resultant messages were judged to have the same clarity as the authored semi-automated messages written by Captain Wolven (Wolven 1991).

The sample automated messages require less than half the authorship of the sample semi-automated messages. It is estimated that there will be a fivefold decrease in the authorship of all the automated messages as compared to the semi-automated messages.

My experience with authoring sentence structures and dictionaries convinces me that automated Dynamic Help messages are much easier to change and therefore more flexible when the program or the needs of the users change. The claim made by Dr. Young (Young 1990a) that a Dynamic Help system is partially immune to changes in the supported program, whereas an ordinary help system would need to be rewritten, appears to be justified.

These conclusions are linked to the objected-oriented nature of the relational databases (Informix and ACE) used to design ACIFS. Similar results can be expected for any program that was designed by either a database (relational or non-relational) or object-oriented language (e.g. C++, Pascal OOP). User support in the form of Dynamic Help can be written and tested in less time and with less authorship than either totally rewriting an existing package or rewriting or changing its user interface.

5.2 Recommendations

5.2.1 Application of Dynamic Help to ISM

I recommend that the Dynamic Help generator shown in Figure 5-1 be used to implement user support to the entire family of DBMS-based or object-oriented ISM packages.

5.2.2 Dynamic Help Generator

The automation of Dynamic Help could be facilitated by a set of software developer's tools that could be called a Dynamic Help Generator. An ad-hoc Dynamic Help Generator was already under development at AIRMICS upon completion of the present study (Washechek, 1991).

If the method proposed in Chapter III is followed in providing Dynamic Help for an existing program, the software developer will be responsible for designing and implementing several software products to be added to the existing program. These are listed in Figure 5-1.

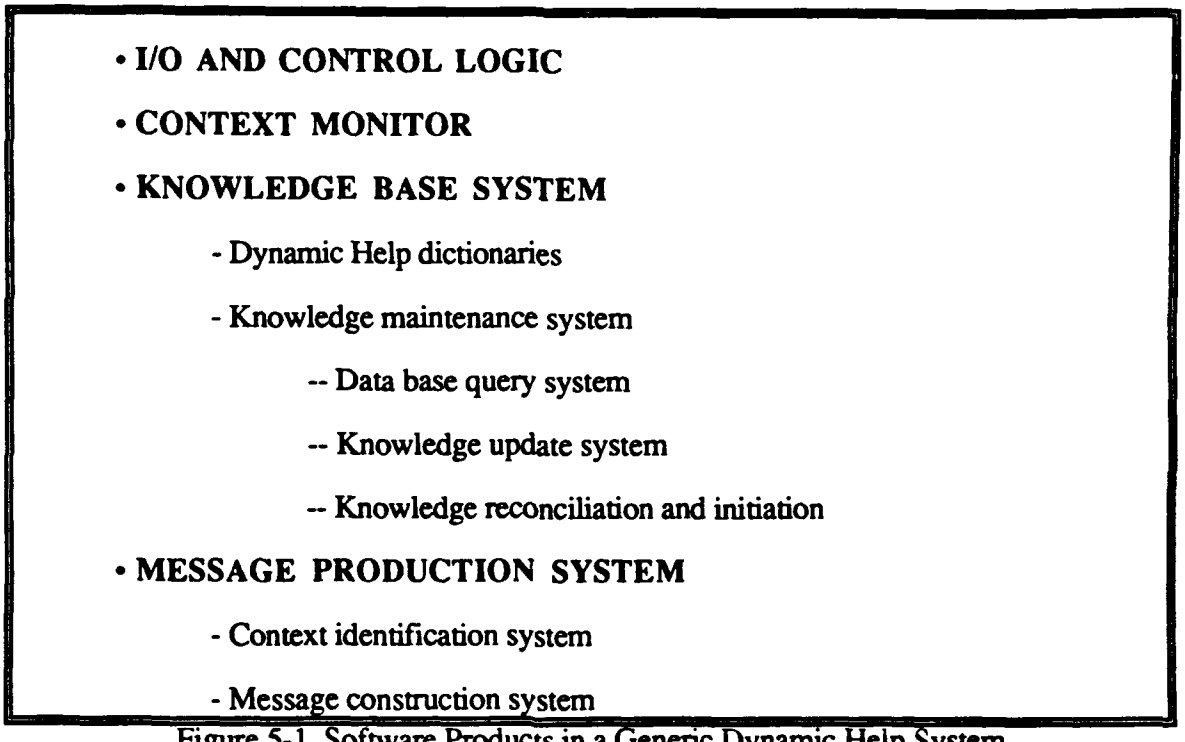


Figure 5-1 Software Products in a Generic Dynamic Help System

I/O control logic must be provided to allow Dynamic Help to be invoked at run time, to allow messages to be displayed, and to allow for return to the context and state from which Dynamic Help was invoked. The design and implementation of I/O and control logic depends entirely on the existing program, not on Dynamic Help design.

Typically the context monitor would consist of code added to the screen control system of the existing program, or to individual forms logic or trigger logic if the interface

design follows a DBMS forms utility. The main job of the context monitor is to translate the existing program's context tracking into values of the active acts and active objects in the Dynamic Help context vector. This translation must be performed, or at least triggered, from wherever the existing program alters or tracks context variables. In a well-designed object-oriented program some of the context values will already be available; for example, in ACIFS the identifier of the current screen is a global variable, so this aspect of context monitoring is already provided by the existing program.

The Dynamic Help dictionaries, as designed in Chapter III and illustrated in Chapter IV, have a structure that is mostly independent of the data structure of the existing program. This means that there may be an opportunity for a Dynamic Help Generator to provide software developer's tools to aid in defining the structure and contents of the dictionaries.

Figure 5-1 lists a "knowledge maintenance system", a system that may be required when low-level objects are database objects. For example, in ACIFS one can add soldiers or clothing items to the database, and attributes of these added soldiers or clothing items can be required for subsequent Dynamic Help messages.

One response to this possibility is to provide for queries to the database, either routinely or only when the dictionaries fail to yield a definition. For DBMS-based programs, such queries are very easy to include in the design, but of course they cause delays because they require disk accesses. Another response is to provide for updating of the dictionaries whenever a new object is added; since the dictionaries have mostly standard structures, a Dynamic Help Generator could include most of the necessary code for this. For systems that may be updated by more than one user, it may be necessary to design a system for reconciliation of the knowledge base with the database. This system would be able to provide the initial data for the knowledge base, as well. Thus a natural task for the Dynamic Help Generator is to provide tools for designing knowledge reconciliation and

initialization of the dictionaries. (Reconciliations should probably occur at session initiations.)

The message production system, including both context identification and message construction, would be mostly independent of the data structure of the existing program. Thus it can be programmed largely "in advance" in the Dynamic Help Generator.

From the foregoing review of the software products that constitute a Dynamic Help system, we conclude that a Dynamic Help Generator can provide tools for designing the dictionaries, the knowledge maintenance system, the context identification system, and the message construction system. I/O and control logic — that is, fitting the Dynamic Help into the existing software system — would be beyond the scope of a Dynamic Help Generator. (context monitoring is also beyond the scope, it is implemented throughout the application program.)

5.2.3 Dynamic Tutorial System

It is recommended that the Dynamic Tutorial system discussed in Section 1.5.3 be implemented to substitute for sending out a user training team each time a new software package is fielded on an installation. Software packages that included Dynamic Help should be sufficiently usable so that a standard teaching tutorial would not be needed, but only a less expensive Dynamic Tutorial system.

APPENDIX 1

SAMPLE DYNAMIC HELP MESSAGES FROM ACIFS

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
	Task 11, posting a manual clothing turn in.	
1	**CLOTHING TURN IN PROCESS** Ready to perform a complete or partial turn in, print Clothing Record (DA 3645), or perform a direct exchange of sized items for an individual who has completed initial issue. To go to Clothing Turn In Process, press Return Key.	**CLOTHING TURN IN PROCESS** Ready to go to Clothing Turn In Process Menu. Clothing turn in processes include: a complete or partial turn in, print clothing record, or perform a direct exchange of sized items for an individual who has completed initial issue. To see general documentation, press F1.
2	**CLOTHING TURN IN PROCESS** Ready to perform a complete or partial turn in, print Clothing Record (DA 3645), or perform a direct exchange of sized items for an individual who has completed initial issue. **Turn In Process** Ready to process a turn in of issued items. Do not use this option if an individual has not completed the issue cycle or had completed a turn in already.	**Turn In Process** Ready to go to Turn In Process. The turn in process will process a turn in of issued items. Do not use this option if an individual has not completed the issue cycle or had completed a turn in already. To see general documentation, press F1.
3	**TURN INS** Ready to process turn in of issued items for an individual. To return to CIF Clothing Turn Ins Menu, press DEL (Delete) Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **SSN** Ready to enter the individual's SSN. Examples: 123456789 for U.S. Soldier K23456789 for KATUSA KC3456789 for Korean Civilian To finish entering SSN, press Return Key.	**SSN** Ready to enter social security number of soldier. Examples: 123456789 for U.S. Soldier K23456789 for KATUSA KC3456789 for Korean Civilian To see general documentation, press F1.
4	**MANUAL TURN INS** Ready to enter information on an individual with a turn in. Complete all entries and press Return Key to continue the turn in process, or press DEL (delete) Key to start another Turn In or to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **Last Name** Ready to enter the individual's Last Name. To go to First Name, enter the Last Name and press Return Key. To finish entering Last Name, press Return Key.	**Last Name** Ready to enter last name of <SSN>. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
5	**MANUAL TURN INS** Ready to enter information on an individual with a turn in. Complete all entries and press Return Key to continue the turn in process, or press DEL (delete) Key to start another Turn In or to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **First Name** Ready to enter the individual's First Name. To go to Sex, enter the First Name and press Return Key. To finish entering First Name, press Return.	**First Name** Ready to enter first name of <SSN>. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. To see general documentation, press F1.
6	**MANUAL TURN INS** Ready to enter information on an individual with a turn in. Complete all entries and press Return Key to continue the turn in process, or press DEL (delete) Key to start another Turn In or to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **Sex** Ready to enter the individual's Sex. To go to Unit Identification Code, enter the Sex and press Return Key. Acceptable entries are M for Male and F for Female. To finish entering Sex, press Return Key.	**Sex** Ready to enter sex of <SSN>. Acceptable entries are M for Male and F for Female. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
7	**MANUAL TURN INS** Ready to enter information on an individual with a turn in. Complete all entries and press Return Key to continue the turn in process, or press DEL (delete) Key to start another Turn In or to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **Unit Identification Code** Ready to enter an individual's Unit Identification Code (UIC). To go to Grade, enter the UIC and press Return Key. The UIC is a code identifying the unit to which an individual is assigned. The UIC may be found on the individual's DA 3645 (manual) or unit assignment order. Example: WE0QAA W134AA WRMAAA (UIC with largest assg. strength) Note: O (letter) and 0 (number) are distinct. To finish entering UIC, press Return Key	**Unit Identification Code** Ready to enter unit identification code of <SSN> The UIC is a code identifying the unit to which an individual is assigned. The UIC may be found on the individual's DA 3645 (manual) or unit assignment order. Example: WE0QAA W134AA WRMAAA (UIC with largest assg. strength) Note: O (letter) and 0 (number) are distinct. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. To see general documentation, press F1.
8	**MANUAL TURN INS** Ready to enter information on an individual with a turn in. Complete all entries and press Return Key to continue the turn in process, or press DEL (delete) Key to start another Turn In or to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **Grade** Ready to enter the Individual's Grade. To go to Duty MOS, enter the Grade and press Return Key. Enlisted - E01 - E10 Civilian - G01 - G15 Officer - O01 - O10 Korean - K01 - K15 Warrant - W01 - W05 Note: O (letter) and 0 (number) are distinct. To finish entering Grade, press Return Key.	**Grade** Ready to enter grade of <SSN> Enlisted - E01 - E10 Civilian - G01 - G15 Officer - O01 - O10 Korean - K01 - K15 Warrant - W01 - W05 Note: O (letter) and 0 (number) are distinct. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
9	**MANUAL TURN INS** Ready to enter information on an individual with a turn in. Complete all entries and press Return Key to continue the turn in process, or press DEL (delete) Key to start another Turn In or to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **Duty MOS** Ready to enter the Individual's Duty MOS. To go to Special Issue, enter the Duty MOS and press Return Key. Examples: 11B20 - infantry (E5) 95B - military police 92B - supply officer To finish entering Duty MOS, press Return Key.	**Duty MOS** Ready to enter duty MOS of <SSN> Examples: 11B20 - infantry (E5) 95B - military police 92B - supply officer NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. To see general documentation, press F1.
10	**MANUAL TURN INS** Ready to enter information on an individual with a turn in. Complete all entries and press Return Key to continue the turn in process, or press DEL (delete) Key to start another Turn In or to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. Do not use this option if the individual has not completed the issue cycle or has already completed turn in. **Special Issue?** Ready to determine whether an individual has a Duty MOS which required a special issue. Acceptable entries are Y and N. A special issue is authorized for the following Duty MOS's: 95B 15A 94B 67U 63B If the individual's Duty MOS is listed above, enter Y and press Return Key. Otherwise, enter N and press Return Key.	**Special Issue?** Ready to enter special issue status of <SSN>. Acceptable entries are Y and N. A special issue is authorized for the following Duty MOS's: 95B 15A 94B 67U 63B If the individual's Duty MOS is listed above, enter Y and press Return Key. Otherwise, enter N and press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
11	**MANUAL TURN IN - ITEMS RETURNED** Ready to enter the sizes and quantities turned in for each LIN displayed. If a soldier was not signed for a listed LIN, use the F2 Key to delete it. To add a LIN, press the F1 key while on the LIN which alphanumerically follows the new LIN. When all entries are completed, press ESC Key on the first blank line. Press DEL (Delete) Key to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. To make a correction, press UP Arrow Key, DOWN Arrow Key,BACKSPACE Key, or Return Key as needed, and perform entry again. **TIQTY** Ready to enter for <LIN.description> the quantity turned in. If the quantity turned in matches the quantity under the TIQTY field, press Return Key. To finish entering TIQTY, press Return Key.	**TIQTY** Ready to enter turn in quantity of <LIN.description>. If the quantity turned in matches the quantity under the TIQTY field, press Return Key. Enter the sizes and quantities turned in for each LIN displayed. If a soldier was not signed for a listed LIN, use the F2 Key to delete it. To add a LIN, press the F1 key while on the LIN which alphanumerically follows the new LIN. When all entries are completed, press ESC Key on the first blank line. Press DEL (Delete) Key to return to Clothing Turn Ins Menu without saving entries. To see general documentation, press F1.
12	**MANUAL TURN IN - ITEMS RETURNED** Ready to enter the sizes and quantities turned in for each LIN displayed. If a soldier was not signed for a listed LIN, use the F2 Key to delete it. To add a LIN, press the F1 key while on the LIN which alphanumerically follows the new LIN. When all entries are completed, press ESC Key on the first blank line. Press DEL (Delete) Key to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. To make a correction, press UP Arrow Key, DOWN Arrow Key,BACKSPACE Key, or Return Key as needed, and perform entry again. **Size** Ready to enter size of <LIN.description> To go to TIQTY, enter the Size and press Return Key. Valid Size entries (Up to 32 values displayed): L M	**Size** Ready to enter size of <LIN.description> Valid Size entries: <LIN.validsize> Enter the sizes and quantities turned in for each LIN displayed. If a soldier was not signed for a listed LIN, use the F2 Key to delete it. To add a LIN, press the F1 key while on the LIN which alphanumerically follows the new LIN. When all entries are completed, press ESC Key on the first blank line. Press DEL (Delete) Key to return to Clothing Turn Ins Menu without saving entries. To see general documentation, press F1.
13	**MANUAL TURN IN - ITEMS RETURNED** Ready to enter the sizes and quantities turned in for each LIN displayed. If a soldier was not signed for a listed LIN, use the F2 Key to delete it. To add a LIN, press the F1 key while on the LIN which alphanumerically follows the new LIN. When all entries are completed, press ESC Key on the first blank line. Press DEL (Delete) Key to return to Clothing Turn Ins Menu without saving entries. To complete an entry, press Return Key. To make a correction, press UP Arrow Key, DOWN Arrow Key,BACKSPACE Key, or Return Key as needed, and perform entry again. **LIN** Ready to enter the LIN number of an item which is part of an addtional issue. Entry should be of the form A#### where A denotes a letter and # denotes a number.	**LIN** Ready to enter line item number of inventory item. Enter the sizes and quantities turned in for each LIN displayed. If a soldier was not signed for a listed LIN, use the F2 Key to delete it. To add a LIN, press the F1 key while on the LIN which alphanumerically follows the new LIN. When all entries are completed, press ESC Key on the first blank line. Press DEL (Delete) Key to return to Clothing Turn Ins Menu without saving entries. Entry should be of the form A#### where A denotes a letter and # denotes a number. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
	Task 12.2, posting a statement of charges.	
14	**ADJUSTMENT TRANSACTIONS** Ready to perform Adjustment Transactions such as: - Statements of Charges (S/C) - Reports of Survey (R/S) - Cash Collection (C/C) - Lateral Transfers - Administrative Adjustments - Found on Installation - Turn Ins to SSA/DRMO - Cancelling S/C, R/S, C/C To go to Adjustment Transactions, press Return Key. To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL (Delete) Key.	**ADJUSTMENT TRANSACTIONS** Ready to go to Adjustment Transactions Menu. Adjustment transactions include: statements of charges, Reports of survey, cash collection, lateral transfers, administrative adjustments, found on installation, turn ins to SSA/DRMO, and cancelling S/C, R/S, C/C. To see general documentation, press F1.
15	**ADJUSTMENT TRANSACTIONS MENU** Ready to choose one of the following types of transactions: Statement of Charges Report of Survey Cash Collection Voucher Lateral Transfer Out Lateral Transfer In Administrative Adjustment Found on Installation Turn In to SSA/DRMO Cancel S/C, R/S, C/C To return to CIF Main Menu, highlight option X and press Return Key. Proceed to the next help screen for more information about menu option 1, Statement of Charges. **Statement of Charges** Ready to enter Statements of Charges. This option subtracts losses from the Property Book on hand quantity and creates a transaction record of the dollar value for the document register. To go to Statement of Charges, press Return Key. To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL (Delete) Key.	**Statement of Charges** Ready to go to Statement of Charges. A statement of charges subtracts losses from the property book on hand quantity and creates a transaction record of the dollar value for the document register. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
16	**STATEMENT OF CHARGES** Ready to enter a Statement of Charges. Up to 100 NSNs are allowed per statement. To return to Adjustment Transactions Menu, press DEL (Delete) Key. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in any of the NSN or Quantity fields, press LEFT Arrow Key, UP Arrow Key, or Return Key to return to the field, perform entry again, and press Return Key. **NSN** Ready to enter the NSN of an item listed on the Statement of Charges. To go to Quantity, enter the NSN and press Return Key. Acceptable entries are zero and positive numbers. Example: 8415001841352 To finish entering NSN, press Return Key. To make a correction while still in the NSN field, use BACKSPACE and over-strike incorrect data.	**NSN** Ready to enter NSN of inventory item. Acceptable entries are zero and positive numbers. Example: 8415001841352 Up to 100 NSNs are allowed per statement. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. To see general documentation, press F1.
17	**STATEMENT OF CHARGES** Ready to enter a Statement of Charges. Up to 100 NSNs are allowed per statement. To return to Adjustment Transactions Menu, press DEL (Delete) Key. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in any of the NSN or Quantity fields, press LEFT Arrow Key, UP Arrow Key, or Return Key to return to the field, perform entry again, and press Return Key. **Quantity** Ready to enter for <LIN.description>the Quantity as listed on the Statement of Charges. To finish entering Quantity, press Return Key. To make a correction while still in the Quantity field, use BACKSPACE and over-strike incorrect data.	**Quantity** Ready to enter quantity of <LIN.description>. Acceptable entries are 1 to 99999. Up to 100 NSNs are allowed per statement. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. To see general documentation, press F1.
18	**STATEMENT OF CHARGES** Ready to enter a Statement of Charges. Up to 100 NSNs are allowed per statement. To return to Adjustment Transactions Menu, press DEL (Delete) Key. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. NOTE: BACKSPACE Key does not erase data. To make a correction in any of the NSN or Quantity fields, press LEFT Arrow Key, UP Arrow Key, or Return Key to return to the field, perform entry again, and press Return Key. **Total Price** Ready to enter the Total Price of the items listed on the Statement of Charges. Acceptable values are .01 to 99999.99. To finish entering Total Price, press Return Key. To make a correction while still in the Total Price field, use BACKSPACE and overstrike incorrect data.	**Total Price** Ready to enter total price of document. Acceptable values are .01 to 99999.99. Up to 100 NSNs are allowed per statement. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
	Task 12.3, modify a report of survey.	
19	**ADJUSTMENT TRANSACTIONS MENU** Ready to choose one of the following types of transactions: Statement of Charges Report of Survey Cash Collection Voucher Lateral Transfer Out Lateral Transfer In Administrative Adjustment Found on Installation Turn In to SSA/DRMO Cancel S/C, R/S, C/C To return to CIF Main Menu, highlight option X and press Return Key. Proceed to the next help screen for more information about menu option 2, Report of Survey. **Report of Survey** Ready to enter Reports of Survey. This option subtracts losses from the Property Book on hand quantity and creates a transaction record of the dollar value for the document register. To go to Report of Survey, press Return Key. To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL (Delete) Key.	**Report of Survey** Ready to go to Report of Survey. A Report of Survey subtracts losses from the property book on hand quantity and creates a transaction record of the dollar value for the document register. To see general documentation, press F1.
20	**REPORT OF SURVEY** Ready to enter a Report of Survey. Up to 100 NSN's are allowed per report. Enter the NSNs and quantities of the items listed. After all NSNs/Quantities have been entered, press ESC key, enter the Total Price listed, and press Return Key. To return to Adjustment Transactions Menu, press DEL (Delete) Key. NOTE: BACKSPACE Key does not erase data. **NSN** Ready to enter the NSN of an item listed on the Report of Survey. To go to Quantity, enter the NSN and press Return Key. Acceptable entries are zero and positive numbers. Example: 8415001841352 To finish entering NSN, press Return Key. To make a correction while still in the NSN field, use BACKSPACE and over-strike incorrect data	**NSN** Ready to enter NSN of inventory item. Acceptable entries are zero and positive numbers. Example: 8415001841352 Up to 100 NSNs are allowed per statement. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
21	**REPORT OF SURVEY** Ready to enter a Report of Survey. Up to 100 NSN's are allowed per report. Enter the NSNs and quantities of the items listed. After all NSNs/Quantities have been entered, press ESC key, enter the Total Price listed, and press Return Key. To return to Adjustment Transactions Menu, press DEL (Delete) Key. NOTE: BACKSPACE Key does not erase data. **Quantity** Ready to enter for <LIN.description> the Quantity as listed on the Report of Survey. To finish entering Quantity, press Return Key. To make a correction while still in the Quantity field, use BACKSPACE and over-strike incorrect data.	**Quantity** Ready to enter quantity of <LIN.description>. Acceptable entries are 1 to 99999. Up to 100 NSNs are allowed per statement. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. To see general documentation, press F1.
22	**REPORT OF SURVEY** Ready to enter a Report of Survey. Up to 100 NSN's are allowed per report. Enter the NSNs and quantities of the items listed. After all NSNs/Quantities have been entered, press ESC key, enter the Total Price listed, and press Return Key. To return to Adjustment Transactions Menu, press DEL (Delete) Key. NOTE: BACKSPACE Key does not erase data. **Total Price** Ready to enter the Total Price of the items listed on the Report of Survey. Acceptable values are .01 to 99999.99. To finish entering Total Price, press Return Key. To make a correction while still in the Total Price field, use BACKSPACE and overstrike incorrect data.	**Total Price** Ready to enter total price of document. Acceptable values are .01 to 99999.99. Up to 100 NSNs are allowed per statement. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
	Task 13.1, create supply requisitions.	
23	**DOCUMENT REGISTER ACTIONS MENU** Ready to perform the following Document Register Actions: - Create Supply Requisitions - Query Document Register - Post Status Changes or Cancellations - Post Receipts - Request Changes, Followups, or Cancellations - Change Stock Number - Print Supply Transactions - Reopen a Closed Supply Document or Requisition - Query Due In Documents by NSN To go to Document Register Actions, press Return Key. To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL	**DOCUMENT REGISTER ACTIONS MENU** Ready to go to Document Register Actions. Document register actions include: create supply requisitions; query document register; post status changes or cancelations; post receipts; request change;, follow-ups, or cancellations; change stock number; print supply transactions; reopen a closed supply document or requisition; query due in documents by NSN. To see general documentation, press F1.
24	**DOCUMENT REGISTER ACTIONS MENU** Ready to create supply requisitions, maintain and query the document register, print supply transactions and create floppy disk, or query due in documents by NSN. To return to CIF Main Menu, highlight option X and press Return Key. **Create Supply Requisitions** Ready to Create Supply Requisitions. The document number and date will be assigned by the system. To go to Create Supply Requisitions, press Return Key. To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL (Delete) Key.	**Create Supply Requisitions** Ready to go to create supply requisitions. The system will assign the document number. Once the NSN, document identifier code, quantity ordered, and unit of issue entries are completed and no other change is desired, press the ESC key. Once the NSN entry is completed, a correction to the NSN cannot be made. Press the DEL (Delete) Key and reenter NSN to make a correction. To see general documentation, press F1.
25	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Stock Number** Ready to enter the Stock Number of a new requisition. To go to Document Identifier Code, enter the Stock Number. Acceptable entries are zero and positive numbers. Example: 8415011841352 Note: Filling the field signals completion, and the cursor jumps to the Document Identifier Code field. To make a correction while still in the Stock Number field, use BACKSPACE and over-strike incorrect data.	**Stock Number** Ready to enter national stock number of inventory item. Acceptable entries are zero and positive numbers. Example: 8415011841352 NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
26	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Document Identifier Code** Ready to enter the Document Identifier Code (DIC) of the supply requisition. NOTE: This entry is required to process the transaction. To go to Quantity Ordered, enter the Document Identifier Code. The DIC consists of 3 letters/digits which identify the requisition type. LIST: A01 - Overseas Shipment with NSN A0A - Domestic Shipment with NSN A05 - Overseas Shipment without NSN A0E - Domestic Shipment without NSN Note: Filling the field signals completion, and cursor jumps to Quantity Ordered field. To make a correction while still in the Document Identifier Code field, use BACKSPACE and overstrike incorrect data.	**Document Identifier Code** Ready to enter document identifier code of <LIN.description>. NOTE: This entry is required to process the transaction. The DIC consists of 3 letters/digits which identify the requisition type. See the displayed list of appropriate document identifier codes. Examples: A01 - Overseas Shipment with NSN A0A - Domestic Shipment with NSN A05 - Overseas Shipment without NSN A0E - Domestic Shipment without NSN NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.
27	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Quantity Ordered** Ready to enter the Quantity Ordered. To return to Document Identifier Code, press UP Arrow Key. To go to Unit of Issue enter the Quantity and press Return Key. Acceptable entries are 1 to 99999. To finish entering Quantity Ordered, press Return Key. To make a correction while still in the Quantity Ordered field, use BACKSPACE and over-strike incorrect data.	**Quantity Ordered** Ready to enter quantity ordered of <LIN.description>. Acceptable entries are 1 to 99999. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
28	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Unit of Issue** Ready to verify or enter the Unit of Issue (UI) of the item to be requested. NOTE: This entry is required to process the transaction. To return to Quantity Ordered, press UP Arrow Key. To go to Media and Status Code, press Return Key. If the UI is not listed, enter the UI for the NSN as specified by the current AMDF. Example: EA - Each PR - Pair Note: Filling the field signals completion, and cursor jumps to Media and Status Code field. To make a correction while still in the Unit of Issue field, use BACKSPACE and over-strike incorrect data.	**Unit of Issue** Ready to enter unit of issue of <LIN.description>. NOTE: This entry is required to process the transaction. If the UI is not listed, enter the UI for the NSN as specified by teh current AMDF. Example: EA - Each PR - Pair NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.
29	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Media and Status Code** Ready to change the Media and Status Code, if applicable. Otherwise, press Return Key. To return to Unit of Issue, press UP Arrow Key. To go to Demand Code, press Return Key. The Media and Status Code indicates whether the requisitioner and/or the DSU should receive status on a document by the supply source. See AR725-50 for specific codes. Note: Filling the field signals completion, and cursor jumps to Demand Code field. To make a correction while while still in the Media and Status Code field, use BACKSPACE and over-strike incorrect data.	**Media and Status Code** Ready to enter media and status code of <LIN.description>. The Media and Status Code indicates whether the requisitioner and/or the DSU should receive status on a document by the supply source. See AR725-50 for specific codes. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
30	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Demand Code** Ready to change the Demand Code, if applicable. Otherwise, press Return Key. NOTE: This entry is required to process the transaction. To return to Media and Status Code, press UP Arrow Key. To go to Signal Code, press Return Key. The Demand Code indicates whether the demand is recurring or nonrecurring. R - Recurring Demand N - Non-recurring demand Note: Filling the field signals completion, and cursor jumps to Signal Code field. To make a correction while while still in the Demand Code field, use BACKSPACE and over-strike incorrect data.	**Demand Code** Ready to enter demand code of <LIN.description>. The Demand Code indicates whether the demand is recurring or nonrecurring. R - Recurring Demand N - Non-recurring demand See the displayed list of appropriate demand codes. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.
31	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Signal Code** Ready to change the Signal Code, if applicable. Otherwise, press Return Key. To return to Demand Code, press UP Arrow Key. To go to End Item Code, press Return Key. The Signal Code designates the intended consignee and the activity to receive/pay bills. See AR725-50 for specific codes. Note: Filling the field signals completion, and cursor jumps to End Item Code field. To make a correction while still in the Signal Code field, use BACKSPACE and over-strike incorrect data.	**Signal Code** Ready to enter signal code of <LIN.description>. The Signal Code designates the intended consignee and the activity to receive/pay bills. See AR725-50 for specific codes. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
32	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **End Item Code** Ready to enter the End Item Code. This is an optional entry field. To return to Signal Code, press UP Arrow Key. To go to Priority Code, press Return Key. The End Item Code informs the supply system about the major end item required. Note: Filling the field signals completion, and cursor jumps to Priority Code field. To make a correction while still in the End Item Code field, use BACKSPACE and over-strike incorrect data.	**End Item Code** Ready to enter end item code of <LIN.description>. The End Item Code informs the supply system about the major end item required. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.
33	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Priority Code** Ready to change the Priority Code, if applicable. Otherwise, press Return Key. NOTE: This entry is required to process the transaction. To return to End Item Code, press UP Arrow Key. To go to Required Delivery Date, press Return Key. The Priority Code is a combination of the Force Activity Designator and the Urgency Need Designator. Note: Filling the field signals completion, and cursor jumps to Required Delivery Date field. To make a correction while still in the Priority Code field, use BACKSPACE and over-strike incorrect data.	**Priority Code** Ready to enter priority code of <LIN.description>. The Priority Code is a combination of the Force Activity Designator and the Urgency Need Designator. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
34	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Required Delivery Date** Ready to enter the Required Delivery Date, if applicable. Otherwise, press Return Key. To return to Priority Code, press UP Arrow Key. To go to Advice Code, press Return Key. Required Delivery Date is the date on which the item must be delivered to the CIF. Format is (YDDD). Example: 5 Jan 1991 = 1005. To cancel this entry, over-strike any entries with SPACE bar. Note: Filling the field signals completion, and cursor jumps to Advice Code field. To make a correction while still in the Required Delivery Date field, use BACKSPACE and over-strike incorrect data.	**Required Delivery Date** Ready to enter required delivery date of <LIN.description>. Required Delivery Date is the date on which the item must be delivered to the CIF. Format is (YDDD). Example: 5 Jan 1991 = 1005. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.
35	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Advice Code** Ready to enter the Advice Code for a new requisition, if applicable. Otherwise, press Return Key. To return to Required Delivery Date, press UP Arrow Key. To go to Supplementary Address or Location, press Return Key. The Advice Code provides additional information about the requisition when necessary. See AR725-50 for specific codes. Entry should be of the form A#, where A denotes a letter, and # denotes a number. To cancel entry, over-strike any entries with SPACE bar. Note: Filling the field signals completion, and cursor jumps to Supplementary Address or Location field.	**Advice Code** Ready to enter advice code of <LIN.description>. The Advice Code provides additional information about the requisition when necessary. See AR725-50 for specific codes. Entry should be of the form A#, where A denotes a letter, and # denotes a number. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
36	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **Supplementary Address or Location** Ready to change the Supplementary Address or Location, if applicable. Otherwise, press Return Key. To return to Advice Code, press UP Arrow Key. To go to DSU Code, press Return Key. The Supplementary Address or Location indicates the specific account for receiving materiel or documentation. Note: Filling the field signals completion, and cursor jumps to DSU Code field. To make a correction while still in the Supplementary Address or Location field, use BACKSPACE and over-strike incorrect data.	**Supplementary Address or Location** Ready to enter supplementary address of <LIN.description>. The Supplementary Address or Location indicates the specific account for receiving materiel or documentation. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.
37	**CREATE SUPPLY REQUISITIONS** Ready to Create Supply Requisitions. The system will assign the document number. Once the NSN, Document Identifier Code, Quantity Ordered, and Unit of Issue entries are completed and no other change is desired, press the ESC Key. To return to Document Register Actions Menu, press the DEL (Delete) Key. NOTE: Backspace Key does not erase data. NOTE: Once the NSN entry is completed, a correction to the NSN cannot be made. Press DEL (Delete) Key and reenter NSN to make a correction. To make a correction to any other entry field (besides NSN), press UP Arrow Key or press DOWN Arrow Key, perform entry again, and press Return Key. **DSU Code** Ready to change the DSU Code, if applicable. Otherwise, press Return Key. To return to Supplementary Address or Location, press UP Arrow Key. The DSU Code indicates the direct support unit which will process documentation and materiel. To finish entering DSU Code, press Return Key. To make a correction while still in the DSU Code field, use BACKSPACE and over-strike incorrect data.	**DSU Code** Ready to enter DSU code of <LIN.description>. The DSU Code indicates the direct support unit which will process documentation and materiel. NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
	Task 15.1, post a completed clothing issue.	
38	**CLOTHING ISSUES PROCESS** Ready to select one of the following Issue Processes: - Initial Issue Reception - Complete Initial Issue - Print Clothing Record (3645) - Issue Due Out Items - Print Worksheet - Issue Additional Items - Automate a Manual Record To go to Clothing Issues Process, press Return Key. To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL (Delete) Key.	**CLOTHING ISSUES PROCESS** Ready to go to Clothing Issues Process. Clothing Issues processes include: initial issue reception; complete initial issue; print clothing record (3645); issue due out items; print worksheet; issue additional items; automate a manual record. To see general documentation, press F1.
39	**CLOTHING ISSUES** Ready to begin initial issue, complete initial issue, issue due-out or additional items, print clothing record (3645) or worksheet, or automate manual clothing records. To return to CIF Main Menu, highlight option X and press Return Key.Complete **Complete Initial Issue** Ready to enter the size and quantity issued to the individual for every LIN on his clothing worksheet. To go to Complete Initial Issue, press Return Key. To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL (Delete) Key.	**Initial Issue** Ready to go to complete initial issue. Complete all entries and press ESC key on the first blank row to process the entries. NOTE: The fastest way to proceed to an item where a size entry is required is to press the Return key repeatedly until a beep sounds. To see general documentation, press F1.
40	**COMPLETE INITIAL ISSUE** Ready to complete an individual's initial issue. To return to CIF Clothing Issues without saving entries, press DEL (Delete) Key. Complete all entries, and press ESC Key on the first lank row to process the entries. To complete an entry, press Return Key. NOTE: The fastest way to proceed to an item where a Size entry is required is to press the Return Key repeatedly until a beep sounds. To make a correction in an entry field, press UP or DOWN Arrow Keys, BACKSPACE Key, or Return Key to return to the field, and perform entry again. **SSN** Ready to input an individual's SSN. Examples: 123456789 for U.S. Soldier K23456789 for KATUSA KC3456789 for Korean Civilian To finish entering SSN, press Return Key.	**SSN** Ready to enter social security number of soldier. Examples: 123456789 for U.S. Soldier K23456789 for KATUSA KC3456789 for Korean Civilian To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
41	**COMPLETE INITIAL ISSUE** Ready to complete an individual's initial issue. To return to CIF Clothing Issues without saving entries, press DEL (Delete) Key. Complete all entries, and press ESC Key on the first lank row to process the entries. To complete an entry, press Return Key. NOTE: The fastest way to proceed to an item where a Size entry is required is to press the Return Key repeatedly until a beep sounds. To make a correction in an entry field, press UP or DOWN Arrow Keys, BACKSPACE Key, or Return Key to return to the field, and perform entry again. **IQTY** Ready to enter for <LIN.description> the quantity of an item issued to the individual as annotated on his clothing worksheet. If the quantity issued matches the quantity under IQTY, press Return Key. Acceptable entries are zero and positive numbers. To finish entering IQTY, press Return Key.	**IQTY** Ready to enter issued quantity of <LIN.description>. If the quantity issued matches the quantity under the IQTY field, press Return Key. Acceptable entries are zero and positive numbers. To see general documentation, press F1.
42	**COMPLETE INITIAL ISSUE** Ready to complete an individual's initial issue. To return to CIF Clothing Issues without saving entries, press DEL (Delete) Key. Complete all entries, and press ESC Key on the first lank row to process the entries. To complete an entry, press Return Key. NOTE: The fastest way to proceed to an item where a Size entry is required is to press the Return Key repeatedly until a beep sounds. To make a correction in an entry field, press UP or DOWN Arrow Keys, BACKSPACE Key, or Return Key to return to the field, and perform entry again. **Size** Ready to enter size of <LIN.description> issued to individual as indicated by the clothing worksheet. To go to IQTY, enter Size and press Return Key. Valid Size entries (Up to 32 values displayed): To finish entering Size, press Return Key.	**Size** Ready to enter size of <LIN.description>. Valid Size entries (Up to 32 values displayed): <LIN.validsize> To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
	Task 15.1, post an issue of due-out clothing items.	
43	**CLOTHING ISSUES** Ready to begin initial issue, complete initial issue, issue due-out or additional items, print clothing record (3645) or worksheet, or automate manual clothing records. To return to CIF Main Menu, highlight option X and press Return Key. **Issue Additional Items** Ready to issue additional items to an individual. To go to Issue Additional Items, press Return Key. To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL (Delete) Key.	**Issue Additional Items** Ready to go to issue additional items. Complete all entries, and press ESC Key to process the entries. To complete an entry, press Return Key. To see general documentation, press F1.
44	**ISSUE ADDITIONAL ITEMS** Ready to issue additional items to an individual. To return to CIF Clothing Issues without saving entries, press DEL (Delete) Key. Complete all entries, and press ESC Key to process the entries. To complete an entry, press Return Key. To make a correction in an entry field, press UP or DOWN Arrow Keys, BACKSPACE Key, or Return Key to return to the field, and perform entry again. **SSN** Ready to input an individual's SSN. Examples: 123456789 for U.S. Soldier K23456789 for KATUSA KC3456789 for Korean Civilian To finish entering SSN, press Return Key.	**SSN** Ready to enter social security number of soldier. Examples: 123456789 for U.S. Soldier K23456789 for KATUSA KC3456789 for Korean Civilian To see general documentation, press F1.
45	**ISSUE ADDITIONAL ITEMS** Ready to issue additional items to an individual. To return to CIF Clothing Issues without saving entries, press DEL (Delete) Key. Complete all entries, and press ESC Key to process the entries. To complete an entry, press Return Key. To make a correction in an entry field, press UP or DOWN Arrow Keys, BACKSPACE Key, or Return Key to return to the field, and perform entry again. **IQTY** Ready to enter for <LIN>descritpion> the quantity of an item that is part of an additional issue. To go to AQTY, enter the IQTY and press Return Key. Acceptable entries are zero and positive numbers. To finish entering IQTY, press Return Key.	**IQTY** Ready to enter issued quantity of <LIN.description>. If the quantity issued matches the quantity under the IQTY field, press Return Key. Acceptable entries are zero and positive numbers. To see general documentation, press F1.

	Semi-Automated Messages for 46 Sample Contexts	Automated Messages for the Same Contexts
46	**ISSUE ADDITIONAL ITEMS** Ready to issue additional items to an individual. To return to CIF Clothing Issues without saving entries, press DEL (Delete) Key. Complete all entries, and press ESC Key to process the entries. To complete an entry, press Return Key. To make a correction in an entry field, press UP or DOWN Arrow Keys, BACKSPACE Key, or Return Key to return to the field, and perform entry again. **Size** Ready to enter size of <LIN.description> as part of an additonal issue to an individual. To go to IQTY, enter Size and press Return Key. Valid Size entries (Up to 32 values displayed): To finish entering Size, press Return Key.	**Size** Ready to enter size of <LIN.description>. Valid Size entries (Up to 32 values displayed): <LIN.validsize> To see general documentation, press F1.

APPENDIX 2

SAMPLE CONTEXTS FROM ACIFS

	Contexts - Acts			
	4-Task	3-Procedure	2-Function	1-Command
1	m-Main Menu			go to
2	m8-Clothing Turn In Process Menu			go to
3	"ditto"	m81-Turn Ins	id soldier	enter
4	"ditto"	m81-Turn Ins	id soldier	enter
5	"ditto"	m81-Turn Ins	id soldier	enter
6	"ditto"	m81-Turn Ins	id soldier	enter
7	"ditto"	m81-Turn Ins	id soldier	enter
8	"ditto"	m81-Turn Ins	id soldier	enter
9	"ditto"	m81-Turn Ins	id soldier	enter
10	"ditto"	m81-Turn Ins	id soldier	enter
11	"ditto"	m81-Turn Ins	id inv item	enter
12	"ditto"	m81-Turn Ins	id inv item	enter
13	"ditto"	m81-Turn Ins	id inv item	enter
14	m-Main Menu			go to
15	m2-Adjustment Transactions			go to
16	"ditto"	m21-S/C	id inv item	enter
17	"ditto"	m21-S/C	id inv item	enter
18	"ditto"	m21-S/C	id inv item	enter
19	m2-Adjustment Transactions			go to
20	"ditto"	m22-R/S	id inv item	enter
21	"ditto"	m22-R/S	id inv item	enter
22	"ditto"	m22-R/S	id inv item	enter
23	m-Main Menu			go to
24	m6-Document Register Actions			go to
25	"ditto"	m61-Create Supply Requisitions	id inv item	enter
26	"ditto"	"ditto"	id inv item	enter
27	"ditto"	"ditto"	id inv item	enter
28	"ditto"	"ditto"	id inv item	enter
29	"ditto"	"ditto"	id inv item	enter
30	"ditto"	"ditto"	id inv item	enter
31	"ditto"	"ditto"	id inv item	enter
32	"ditto"	"ditto"	id inv item	enter
33	"ditto"	"ditto"	id inv item	enter
34	"ditto"	"ditto"	id inv item	enter
35	"ditto"	"ditto"	id inv item	enter
36	"ditto"	"ditto"	id inv item	enter
37	"ditto"	"ditto"	id inv item	enter
38	m-Main Menu			go to
39	m7-Clothing Issues			go to
40	"ditto"	m72-Complete Issue	id soldier	enter
41	"ditto"	"ditto"	id inv item	enter
42	"ditto"	"ditto"	id inv item	enter
43	m7-Clothing Issues			go to
44	"ditto"	m76-Issue Addtional Items	id soldier	enter
45	"ditto"	"ditto"	id inv item	enter
46	"ditto"	"ditto"	id inv item	enter

	Contexts -	Objects			
	Screen	Document	Soldier	Inventory Item	Screen Field
1	Main Menu				menu item 8
2	Clothing Turn Ins				menu item 1
3	Turn Ins				aeb02t.ssn
4	Turn Ins		SSN		aeb02t.lname
5	Turn Ins		SSN		aeb02t.fname
6	Turn Ins		SSN		aeb02t.sex
7	Turn Ins		SSN		aeb02t.uic
8	Turn Ins		SSN		aeb02t.grade
9	Turn Ins		SSN		aeb02t.mosd
10	Turn Ins		SSN		aeb02t.sissue
11	Turn Ins		SSN		aeb01t.lin
12	Turn Ins		SSN	LIN	aeb01t.siz
13	Turn Ins		SSN	LIN	formonly.iqty
14	Main Menu				menu item 2
15	Adjustment Transactions				menu item 1
16	S/C	DOC#			aeb14t.remarks
17	S/C	DOC#		NSN	aeb14t.qty
18	S/C	DOC#			aeb14t.dolval
19	Adjustment Transactions				menu item 1
20	R/S	DOC#			aeb14t.remarks
21	R/S	DOC#		NSN	aeb14t.qty
22	R/S	DOC#			aeb14t.dolval
23	Main Menu				menu item 6
24	Document Register Actions				menu item 1
25	Supply Requisitions				aeb15t.nsn
26	Supply Requisitions			NSN	aeb15t.dic
27	Supply Requisitions			NSN	aeb15t.qty
28	Supply Requisitions			NSN	aeb15t.ui
29	Supply Requisitions			NSN	aeb15t.msstat
30	Supply Requisitions			NSN	aeb15t.demcde
31	Supply Requisitions			NSN	aeb15t.sigcde
32	Supply Requisitions			NSN	aeb15t.eic
33	Supply Requisitions			NSN	aeb15t.pricde
34	Supply Requisitions			NSN	aeb15t.rdd
35	Supply Requisitions			NSN	aeb15t.statcde
36	Supply Requisitions			NSN	aeb15t.supaddr
37	Supply Requisitions			NSN	aeb15t.dsucde
38	Main Menu				menu item 7
39	Complete Initial Issue				menu item 2
40	Complete Issue				aeb02t.ssn
41	Complete Issue				aeb13t.size
42	Complete Issue				aeb13t.iqty
43	Complete Initial Issue				menu item 6
44	Issue Additional Items				aeb02t.ssn
45	Issue Additional Items				aeb13t.size
46	Issue Additional Items				aeb13t.iqty

APPENDIX 3

SAMPLE DATA DICTIONARIES

COMMANDS: Dictionary of Commands (command is the level-1 type of act)

command	verb	prep	darg	iarg
enter	enter	of	FIELDS.short_descriptor (name = CV(screen_field))	FUNCTIONS.darg (function = CV(function))
go to	go to		FIELDS.short_descriptor (name = CV(screen_field))	
respond	respond			

FUNCTIONS: Dictionary of Functions (function is the level-2 type of act)

function	verb	darg
id soldier	identify	"soldier" or soldier.SSN
id inventory item	identify	"inventory item" or LIN.description
id document	identify	

PROCEDURES: Dictionary of Procedures (procedure is the level-3 type of act)

procedure	screen	
Prepare Statement of Charges	m21	
Prepare Report of Survey	m22	
Create Supply Requisition	m61	
Complete Issue	m72	
Issue Additional Items	m76	
Perform Turn Ins	m81	

TASKS: Dictionary of Tasks (Task is the level-4 type of act)

task	screen	
Leave Main Menu	m	
Perform Adjustment Transactions	m2	
Perform Document Register Actions	m6	
Perform Clothing Issues	m7	
Perform Clothing Turn-Ins	m8	

FIELDS: Dictionary of Screen Fields (screen field is a type of object)

name	short descriptor	meaning
aeb01t.lin	line item number	
aeb01t.siz	size	
aeb02t.fname	first name	
aeb02t.grade	grade	
aeb02t.lname	last name	
aeb02t.mosd	duty MOS	
aeb02t.sex	sex	
aeb02t.sissue	special issue status	A special issue is authorized for the following Duty MOS's: 95B, 15A, 94B, 67U, 63B. If the individuals Duty MOS is listed above, enter Y, otherwise enter N.
aeb02t.ssn	social security number	
aeb02t.uic	unit identification code	The UIC is a code identifying the unit to which an individual is assigned. The UIC may be found on the individual's DA 3635 (manual) or unit assignment order.
aeb13t.iqty	issue quantity	If the quantity issued matches the quantity under the IQTY field, press Return Key.
aeb13t.siz	size	
aeb14t.dolval	total price	
aeb14t.qty	quantity	
aeb14t.remarks	NSN	
aeb15t.demcde	demand code	The Demand Code indicates whether the demand is recurring or nonrecurring.
aeb15t.dic	document identifier code	NOTE: This entry is required to process the transaction. The DIC consists of 3 letters/digits which identify the requisition type.
aeb15t.dsucde	DSU code	The DSU Code indicates the direct support unit which will process documentation and materiel.
aeb15t.eic	end item code	The End Item Code informs the supply system about the major end item required.
aeb15t.msstat	media and status code	The Media and Status Code indicates whether the requisitioner and/or the DSU should receive status on a document by the supply source. See AR725-50 for specific codes.
aeb15t.NSN	national stock number	
aeb15t.pricde	priority code	The Priority Code is a combination of the Force Activity Designator and the Urgency Need Designator.
aeb15t.qty	quantity ordered	
aeb15t.rdd	required delivery data	Required Delivery Date is the date on which the item must be delivered to the CIF.
aeb15t.sigcde	signal code	The Signal Code designates the intended consignee and the activity to receive/pay bills. See AR725-50 for specific codes.
aeb15t.statcde	advice code	The Advice Code provides additional information about the requisition when necessary. See AR725-50 for specific codes.
aeb15t.supaddr	supplementary address	The Supplementary Address or Location indicates the specific account for receiving materiel or documentation.
aeb15t.ui	unit of issue	NOTE: This entry is required to process the transaction. If the UI is not listed, enter the UI for the NSN as specified by the current AMDF.
formonly.iqty	turn in quantity	If the quantity turned in matches the quantity under the TIQTY field, press Return Key.

FIELDS: continued

name	short descriptor	meaning
m8	Clothing Turn In Process Menu	Clothing turn in processes include: a complete or partial turn in, print clothing record, or perform a direct exchange of sized items for an individual who has completed initial issue.
m81	Turn In Process	The turn in process will process a turn in of issued items. Do not use this option if an individual has not completed the issue cycle or had completed a turn in already.
m2	Adjustment Transactions Menu	Adjustment transactions include: statements of charges, Reports of survey, cash collection, lateral transfers, administrative adjustments, found on installation, turn ins to SSA/DRMO, and cancelling S/C, R/S, C/C.
m21	Statement of Charges	A statement of charges subtracts losses from the property book on hand quantity and creates a transaction record of the dollar value for the document register.
m22	Report of Survey	A report of survey subtracts losses from the property book on hand quantity and creates a transaction record of the dollar value for the document register.
m6	Document Register Actions	Document register actions include: create supply requisitions; query document register; post status changes or cancelations; post receipts; request change;, follow-ups, or cancellations; change stock number; print supply transactions; reopen a closed supply document or requisition; query due in documents by NSN.
m61	Create Supply Requisition	The system will assign the document number. Once the NSN, document identifier code, quantity ordered, and unit of issue entries are completed and no other change is desired, press the ESC key.
m7	Clothing Issues Process	Clothing Issues processes include: initial issue reception; complete initial issue; print clothing record (3645); issue due out items; print worksheet; issue additional items; automate a manual record.
m72	Complete Initial Issue	Complete all entries and press ESC key on the first blank row to process the entries. NOTE: The fastest way to proceed to an item where a size entry is required is to press the Return key repeatedly until a beep sounds.
m76	Issue Additional Items	Complete all entries, and press ESC Key to process the entries. To complete an entry, press Return Key.

CHOICES: Dictionary of Choices

name	choice
aeb15t.dic	See the displayed list of appropriate DIC entries.
aeb15t.demcde	See the displayed list of appropriate demand code entries.

FORMATS: Dictionary of Formats

name	format

DOMAINS: Dictionary of Domains

name	domain
aeb01t.lin	Entry should be of the form A#### where A denotes a letter and # denotes a number.
aeb02t.grade	Enlisted- E01-E10 Civilian - G01-G15 Officer- O01-O10 Korean - K01-K15 Warrant - W01-W10 Note: O (letter) and 0 (number) are distinct
aeb02t.mosd	Examples: 11B20 - infantry (E5) 95B - military police 92B - supply officer
aeb02t.sex	Acceptable entries are M for Male and F for Female.
aeb02t.ssn	Examples: 1234565789 for U.S. Solder K23456789 for KATUSA KC3456789 for Korean Civilian
aeb02t.uic	Examples: WE0QAA W134AA WRMAAA (UIC with largest assg. strength) Note: O (letter) and 0 (number) are distinct.
aeb13t.iqty	Acceptable entries are 1 to 99999.
aeb13t.siz	Examples: «select uniquevalidsize from aeb13t where LIN = CV(8)»
aeb14t.iqty	Acceptable entries are 1 to 99999.
aeb15t.demcde	Examples: R - Recurring Demand, N - Nonrecurring Demand.
aeb15t.dic	Examples: A01 - Overseas Shipment with NSN A0A - Domestic Shipment with NSN A05 - Overseas Shipment without NSN A0E - Domestic Shipment without NSN
aeb15t.iqty	Acceptable entries are 1 to 99999.
aeb15t.nsn	Acceptable entries are zero and positive numbers.
aeb15t.rdd	Acceptable entries are (YDDD)
aeb15t.statcde	Entry should be of the form A#, where A denotes a letter, and # denotes a number.
aeb15t.ui	Examples: EA - Each, PR - Pair.
aeb15t.qty	Acceptable entries are 1 to 99999.

ADHOC: Ad-Hoc Dictionary

context	Ad Hoc Sentence
,,,, supply requisitions,,,,	NOTE: BACKSPACE Key does not erase data. Filling the field signals completion of the entry and the cursor jumps to the next data entry item.
,, id soldier,, Turn Ins,,,,	NOTE: BACKSPACE Key does not erase data. To make a correction in an entry field, press UP and DOWN Arrow Keys as needed and preform entry again, or press BACKSPACE (non-destructive), overstrike, and press Return Key.
,, id inv item,, Turn Ins,,,,	Enter the sizes and quantities turned in for each LIN displayed. If a soldier was not signed for a listed LIN, use the F2 Key to delete it. To add a LIN, press the F1 key while on the LIN which alphanumerically follows the new LIN. When all entries are completed, press ESC Key on the first blank line. Press DEL (Delete) Key to return to Clothing Turn Ins Menu without saving entries.
,, id inv item,, Complete Issue,,,,	Up to 100 NSNs are allowed per statement. Enter the NSNs and quantities of the items listed. After all NSNs/quantities have been entered, press ESC Key, enter the Total Price listed, and press Return Key.

GENHLP: General Dictionary

context	General message
menu items	To make another selection, use LEFT and RIGHT Arrow Keys to highlight the desired option number and press Return Key. To undo or reverse a selection after it has been pressed, press DEL (Delete) Key.
data items	To commit all data entered on the screen, press the ESC (Escape) Key. To erase all data entered on the screen, press the DEL (Delete) Key.

BIBLIOGRAPHY

Barge, Walter. (1991). Universal Software Documentation via Dynamic Help. Report ASQB-GM-91-028, U.S. Army Information Systems Engineering Command, Fort Huachuca, AZ 85613-5300, June 1991.

Borenstein, Nathaniel (1985). The Design and Evaluation of On-Line Help Systems. Doctoral Dissertation, Carnegie-Mellon University, Pittsburgh, PA.

Brochmann, John R. (1986). Writing Better Computer User Documentation. New York: Wiley.

Cannon, D. (1990). Automated Central Issue Facility [Computer Program]. Atlanta, GA: U.S. Army Software Developmental Center - Atlanta, Fort Gillem, GA.

Dorazio, Patricia (1988). Help Facilities: A Survey of the Literature. Technical Communication, 35, 118-121.

Gould, John D. (1988). How to Design Usable Systems. In Helander, M. (Editor), Handbook of Human-Computer Interaction, Elsevier Science Publishers B. V. North-Holland, 757-789.

Horton, William K. (1990). Designing & Writing Online Documentation. New York: Wiley.

Hurd, John C. (1983) Writing Online Help. In Proceedings of the 30th International Technical Communication Conference. Washington, DC: Society of Technical Communication, W&E 151-154.

Installation Support Modules. (1990). A briefing paper dated 12 September. Office of the Project Manager, U.S. Army Installation Support Modules, Fort Belvior, VA. Photocopied.

Laurel, Brenda, (Ed.). (1990). The Art of Human-Computer Interface Design. Reading, MA: Addison-Wesley.

Nicol, Anne, & Sellen, Abigail. (1990). Building User-centered On-line Help. In B. Laurel (Ed.), The Art of Human-Computer Interface Design (pp. 143-153). Reading, MA: Addison-Wesley.

Norman, Donald, & Craper, Stephen (Editors). (1986). User Centered System Design: New Perspectives on Huyman-Computer Interaction. Hillsdale, NJ: Lawrence Erlbaum Associates.

Wolven, Renée (1991). Effectiveness Testing of Embedded User Support for U.S. Army Installation-Level Software. Masters thesis, Georgia Institute of Technology, Atlanta, GA. Published as report ASQB-GM-91-027, U.S. Army Information Systems Engineering Command, Fort Huachuca, AZ 85613-5300, June 1991.

Young, Donovan, Miller, M. Wayne, Jr. and Coleman, James P., Jr., (1987). Guide to DSS Development, Vol 1. Computer Systems and Technology Division, Electronics and Computer Systems Laboratory, Georgia Tech Research Institute and School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA. Published as report ASQBG-A-88-001, U.S. Army Information Systems Engineering Command, Fort Huachuca, AZ 85613-5300, June 1991.

Young, Donovan (1990a). Embedded User Support for U. S. Army Installation Software. School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205.

Young, Donovan (1990b). Specification of User Requirements and Dynamic Help Systems Standards for EUS project and CIF Conversion. School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0205.

Washechek, Mark. (1991). A Dynamic Help Generator for U.S. Army Installation-Level Software. Unpublished manuscript, Georgia Institute of Technology, Atlanta, GA.